INTRODUCTION

The goal of this project is to leverage the Snowflake data warehouse in conjunction with the Data Build Tool (dbt) to implement Slowly Changing Dimensions (SCD) Types. This effectively handle changing customer data while guaranteeing the correctness of current data and the preservation of historical records. Logstash can be used to ingest CSV files straight into Snowflake tables. The system was configured with Snowflake as the cloud data warehouse and dbt as the transformation and modeling tool. The project's main focus is incremental loading, in which DBT models are made to handle only fresh or updated data, improving performance and cutting down on pointless calculation.

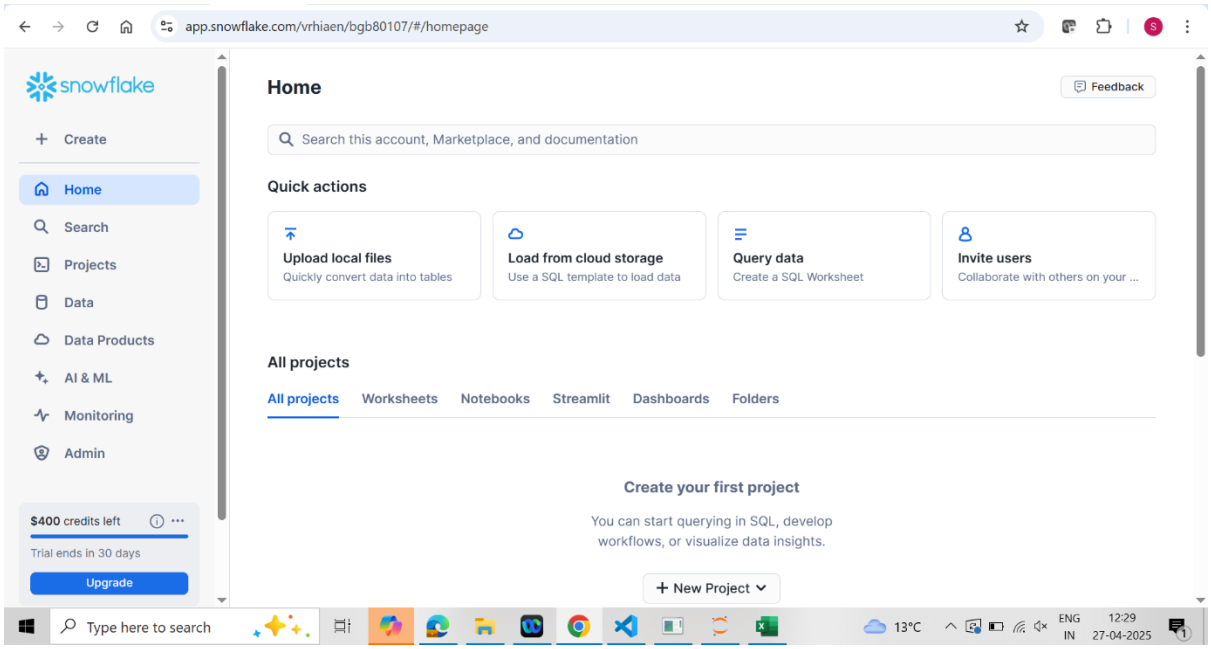
Different business requirements are reflected in the implementation of each form of SCD. When historical tracking is not required, SCD1 offers a simple solution by updating existing entries without keeping any history. By designating old records as inactive and adding new records with updated values, SCD2 maintains the entire history and provides a thorough audit trail of changes over time. By adding new columns, like prev_email, to capture only the most recent change, SCD3 is used to maintain a limited amount of history within a single record. Lastly, by keeping a current table that is frequently updated and a separate history table that records all previous modifications, SCD4 integrates the functionality of SCD1 and SCD2.

To guarantee data quality, the project contains extensive testing and validation using both custom and built-in DBT tests. These include valid flag checks, null value verification, and, when necessary, uniqueness enforcement. To increase performance and maintainability, optimization strategies including incremental model design and modular SQL scripting have also been used. Robust error management and logging are integrated throughout the workflow, guaranteeing dependable data processing and enabling transparent tracking of data concerns.

Environment setup and repository configuration

Snowflake setup

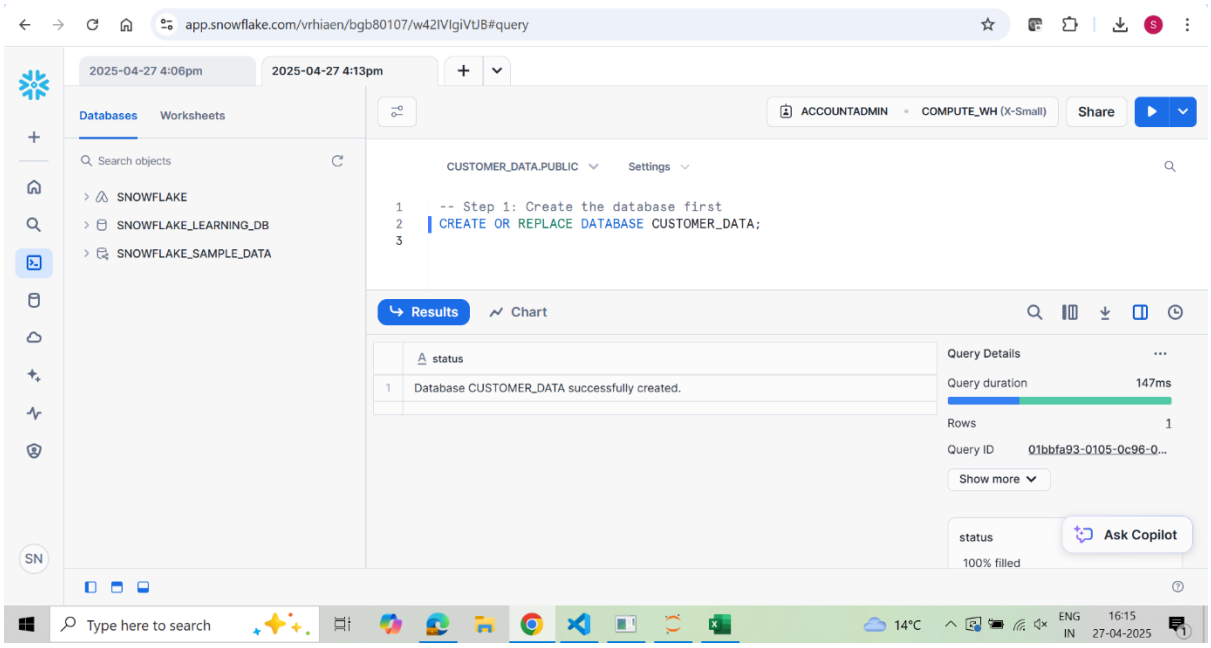
Snowflake account is created as shown in the below screenshot.



Creation of Database

Database called ‘CUSTOMER_DATA’ in snowflake is created by using SQL queries.

CREATE OR REPLACE DATABASE CUSTOMER_DATA;

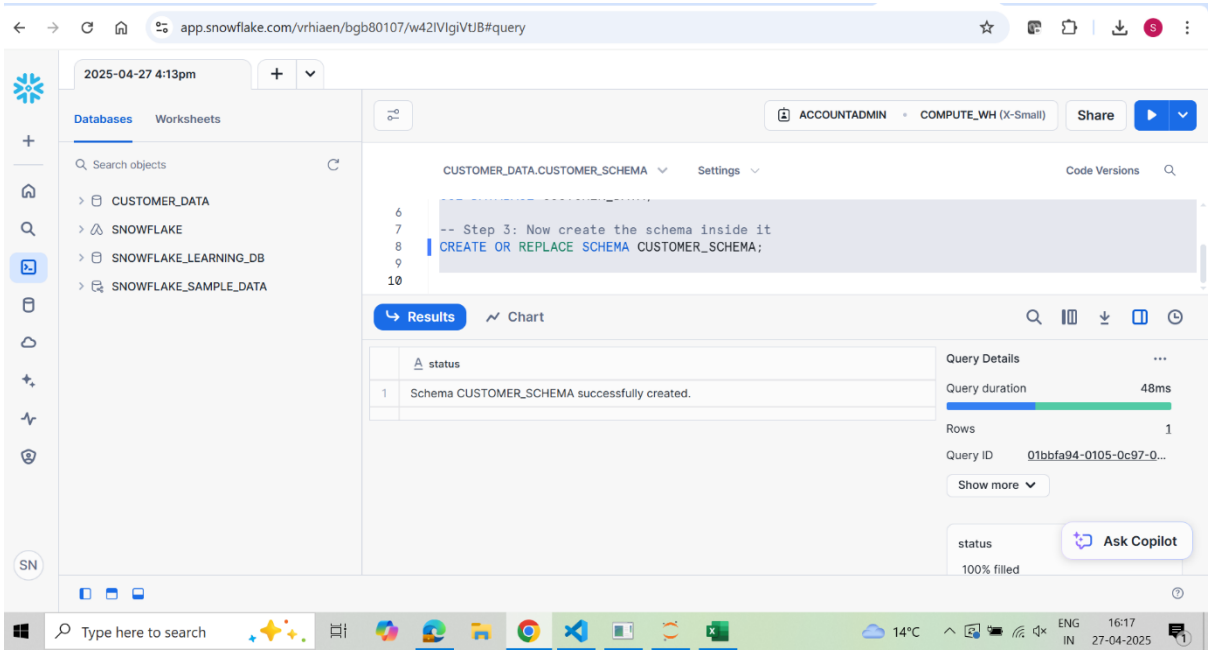


Creation of Schema

Schema called 'CUSTOMER_SCHEMA' is created under 'CUSTOMER_DATA' database by using below queries.

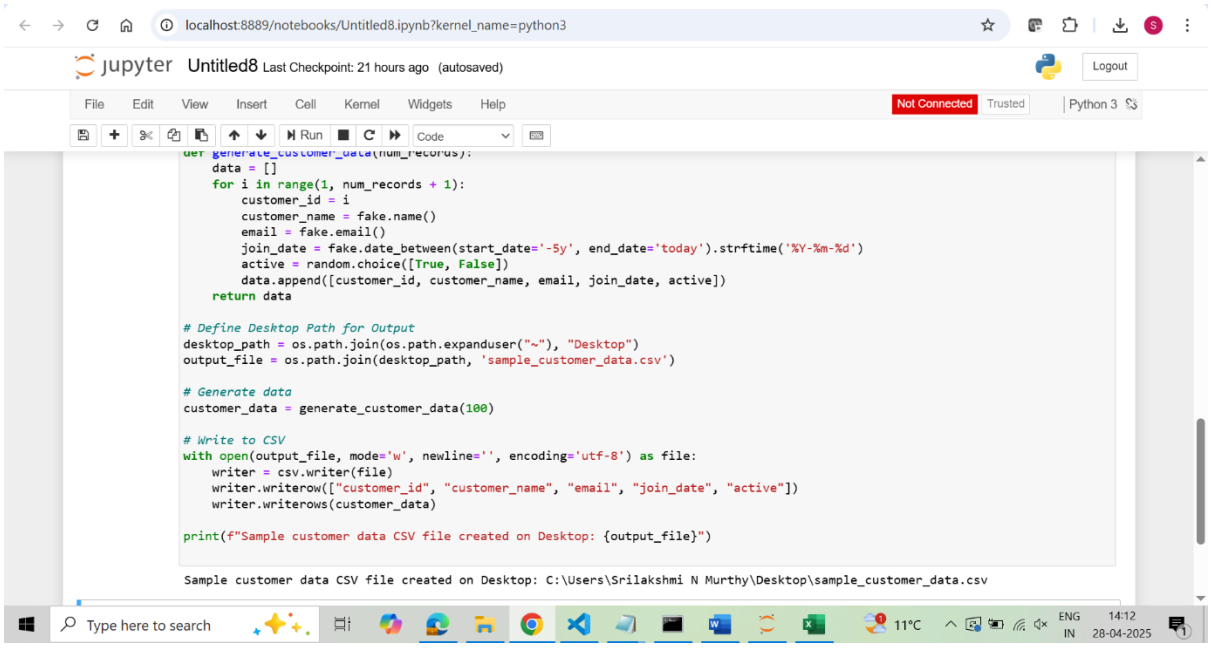
USE DATABASE CUSTOMER_DATA;

CREATE OR REPLACE SCHEMA CUSTOMER_SCHEMA;



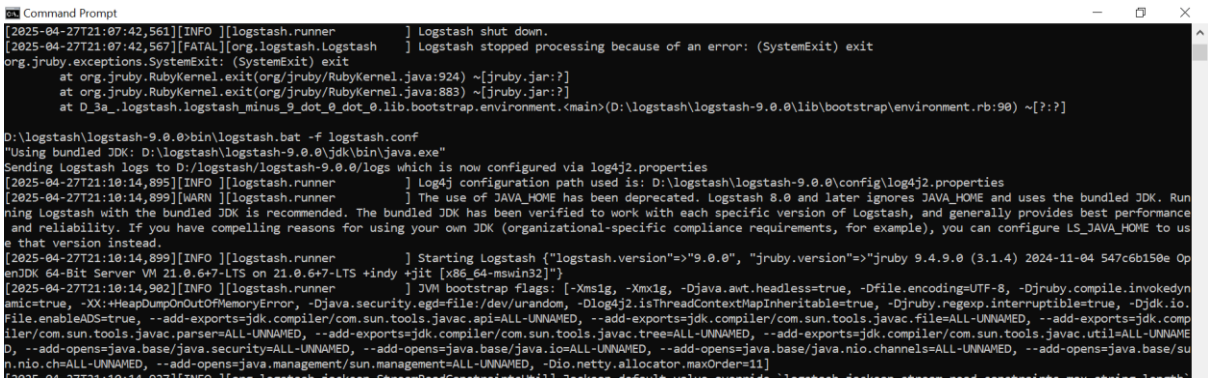
GENERATING CSV WITH FAKER

A csv file called sample_customer_data.csv is generated using faker().

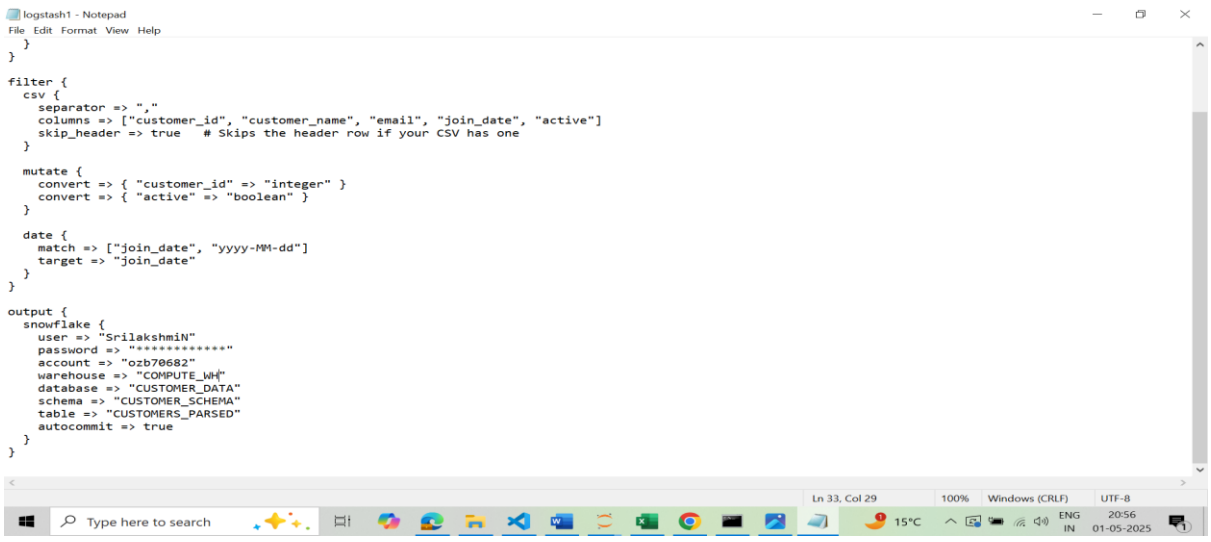


LOGSTASH INSTALLATION

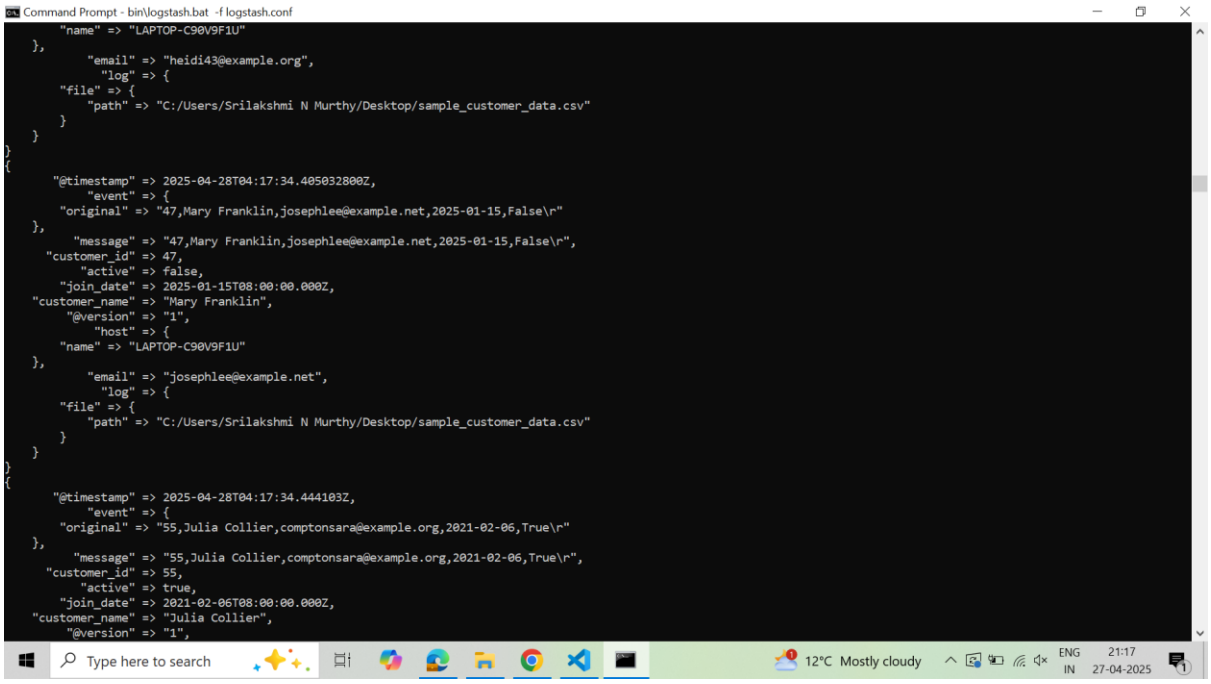
Logstash is used to ingest data into Snowflake. The below screenshot explains about installing Logstash packages.



The below screenshot is about logstash.conf file where we give all the credentials of snowflake like account details, database, schema etc for connection.

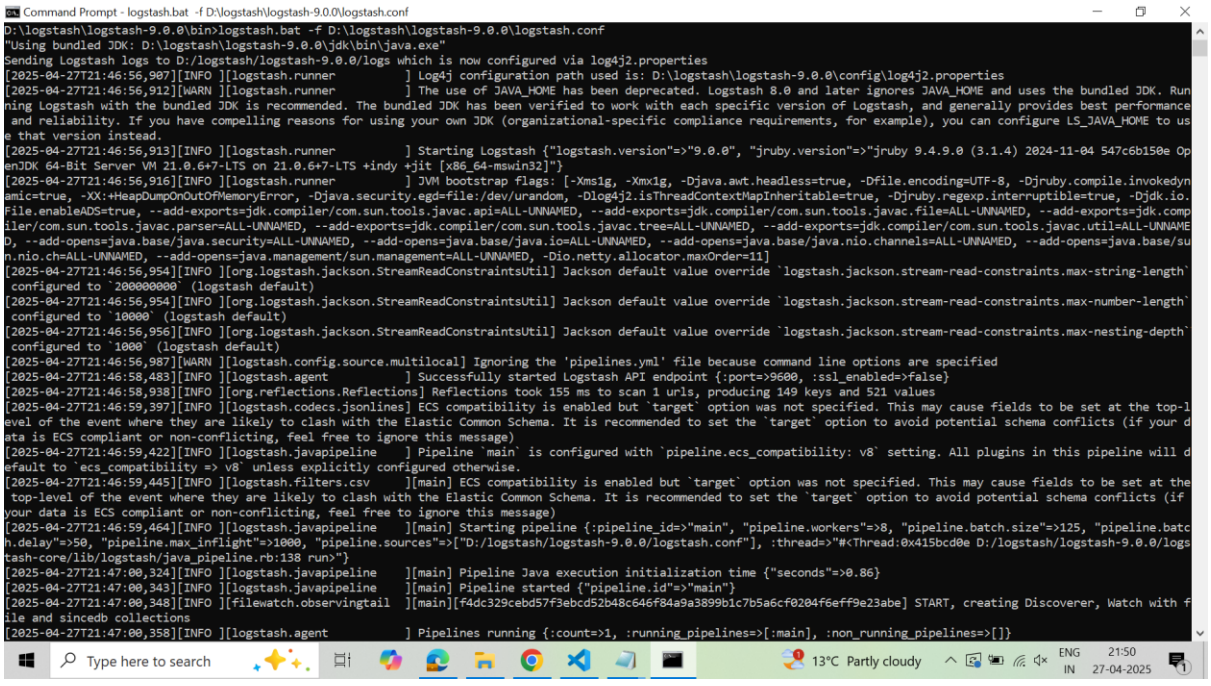


The below code explains the conversion of auto generated csv file into json format through Logstash.



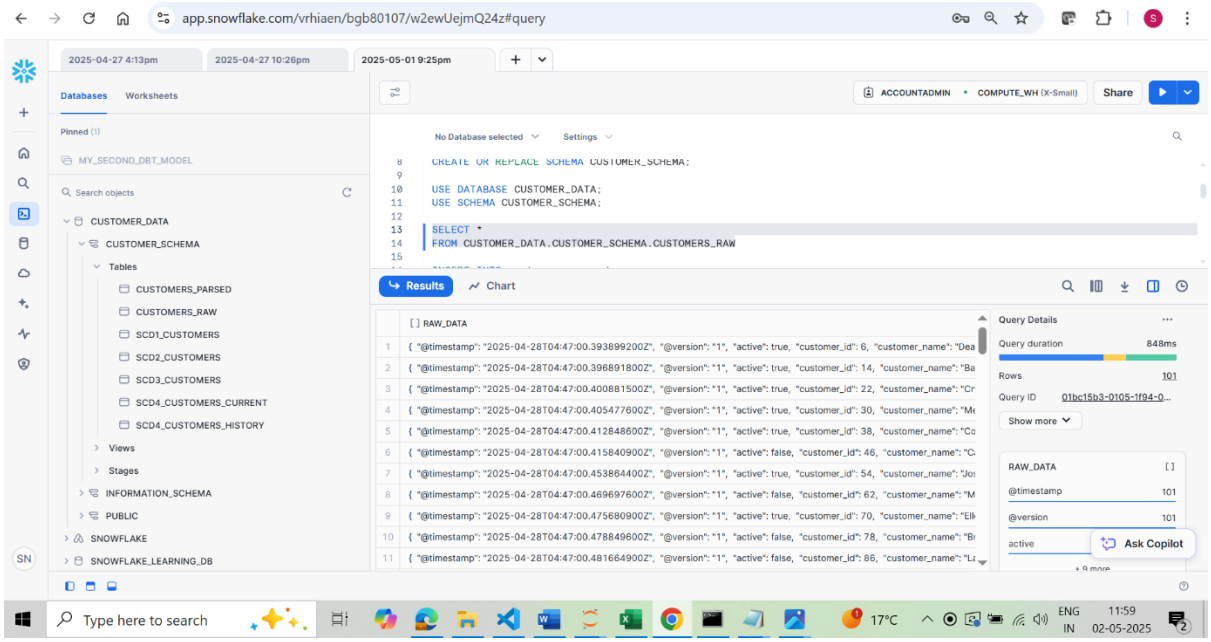
PIPELINE RUNNING

This is the Logstash pipeline where the data is automatically loaded inside Snowflake environment. The below screenshot is about pipeline logs.



When data is loaded into Snowflake from Logstash pipeline, data is loaded in JSON format. Below screenshot shows raw strcture format of data. The raw data is stored in the table called 'CUSTOMERS_RAW'. Snowflake typically stores the JSON data in a Variant Colum i.e raw_data. This means the **whole JSON blob** is in 1 field.

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW



Mapping of raw data into structured data

While Snowflake stores the data as JSON by default, a transformation step is required to extract and insert this data into structured tables for querying. This mapping query ensures that JSON data becomes clean, structured, and usable within the database schema. The table 'customers_raw' contains raw JSON data. The below query reads JSON objects and **maps/extracts each field into a structured format** and insters into 'customers_parsed' table. The below query is casting datatypes to INT, STRING, BOOLEAN.

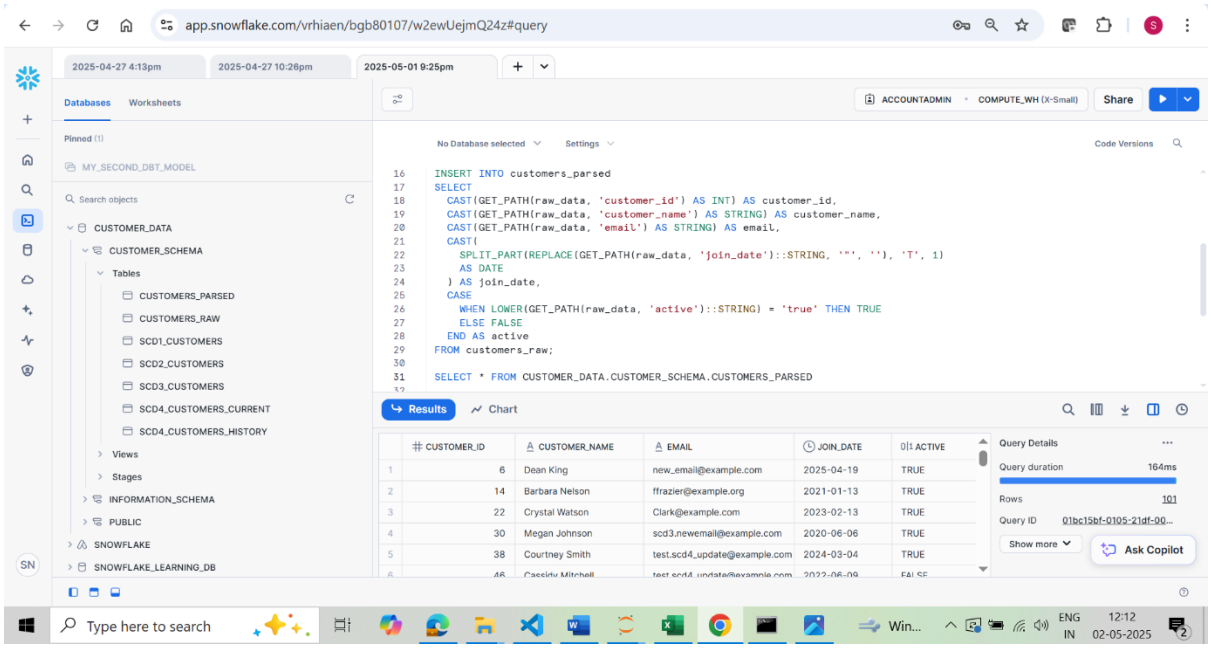
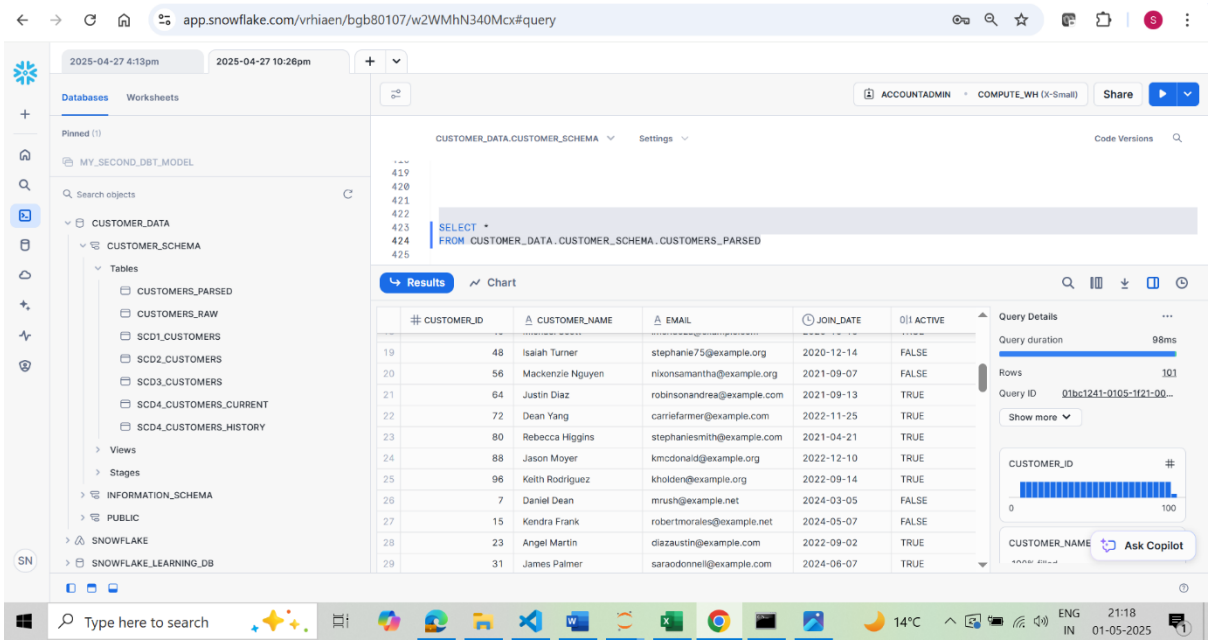


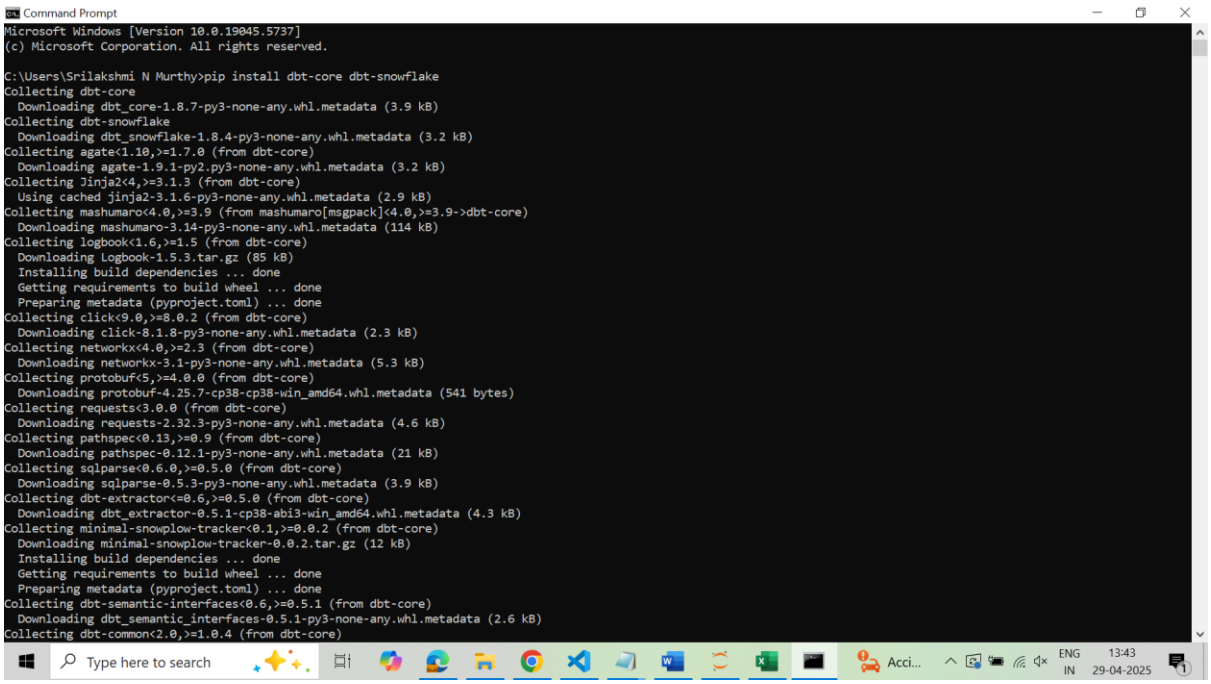
TABLE GENERATION

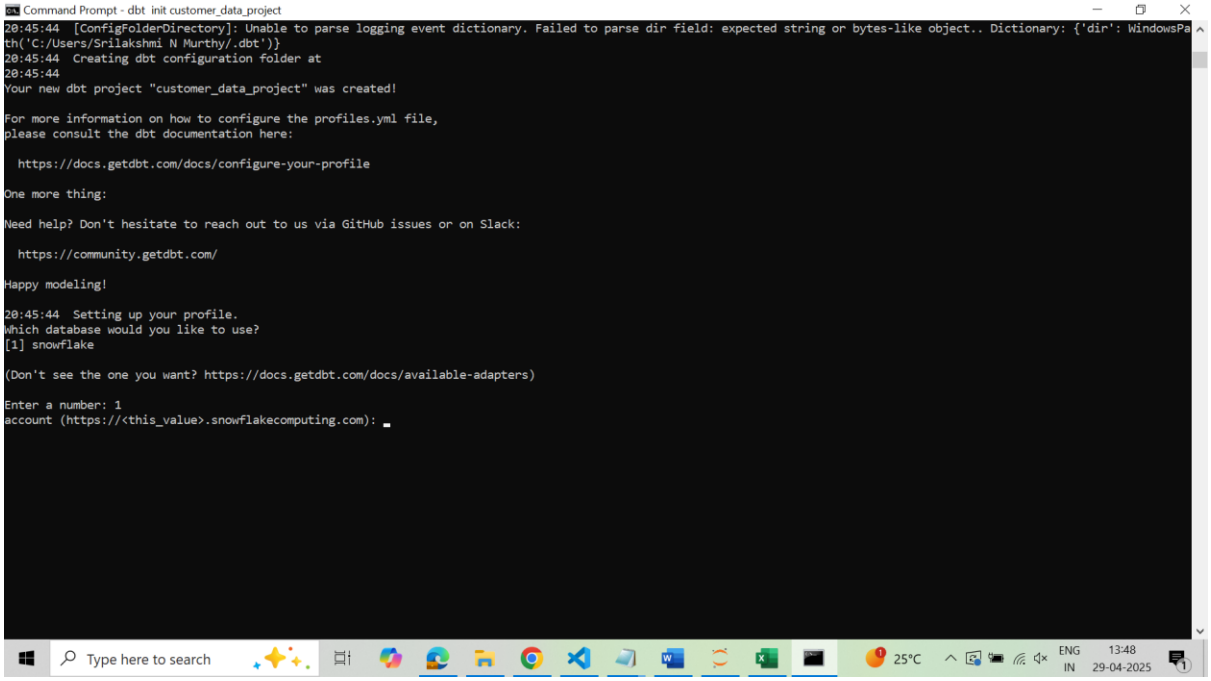
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED



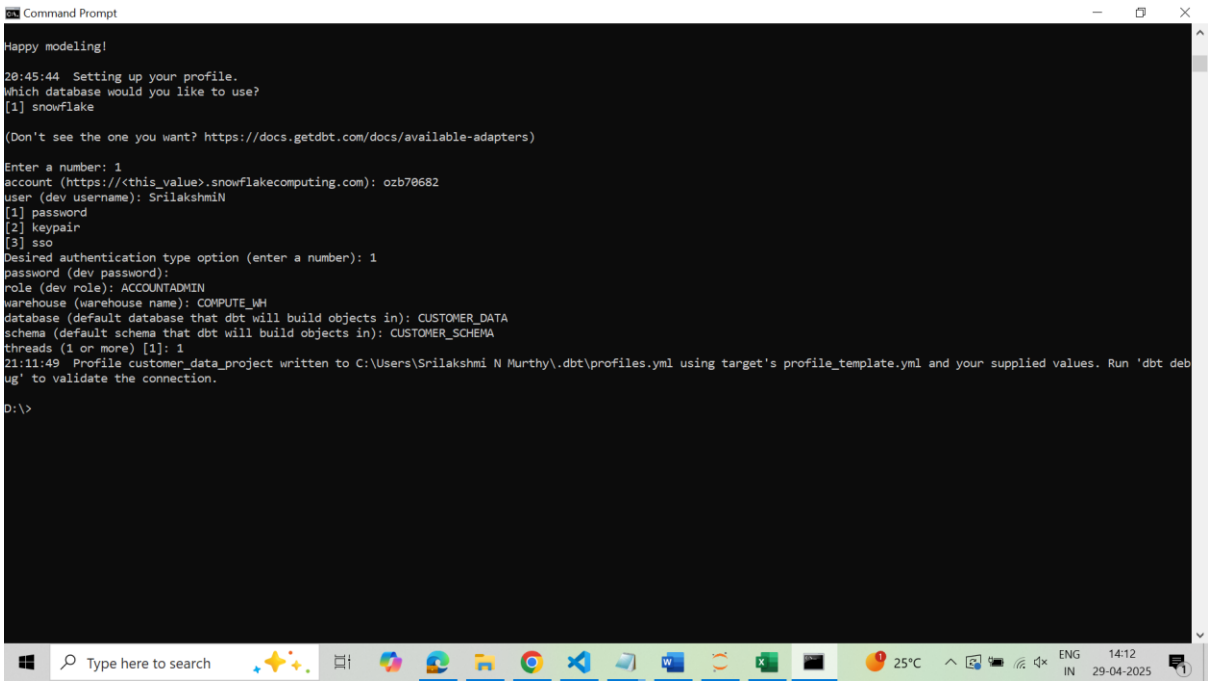
DBT INSTALLATION

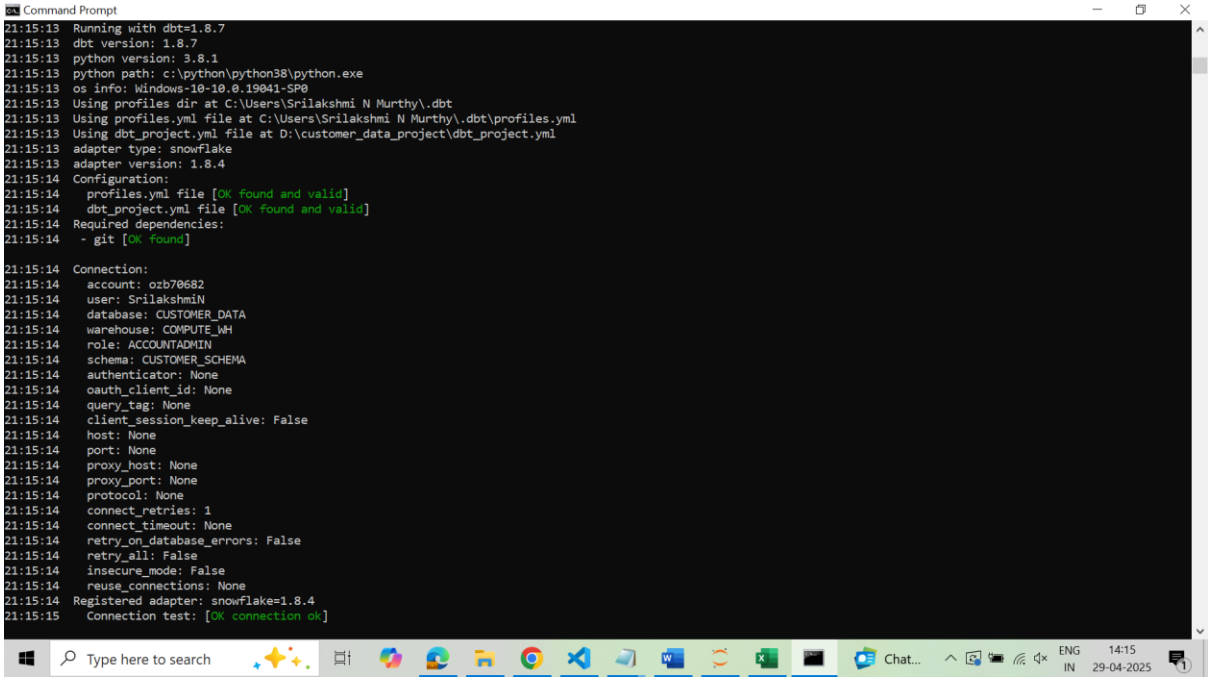
Dbt helps transform raw data into clean, analysable datasets inside data warehouse. dbt supports incremental loading, testing, and documentation, making data pipelines efficient and reliable. With features like ref(), dbt ensures transformations are easy to manage and scalable.

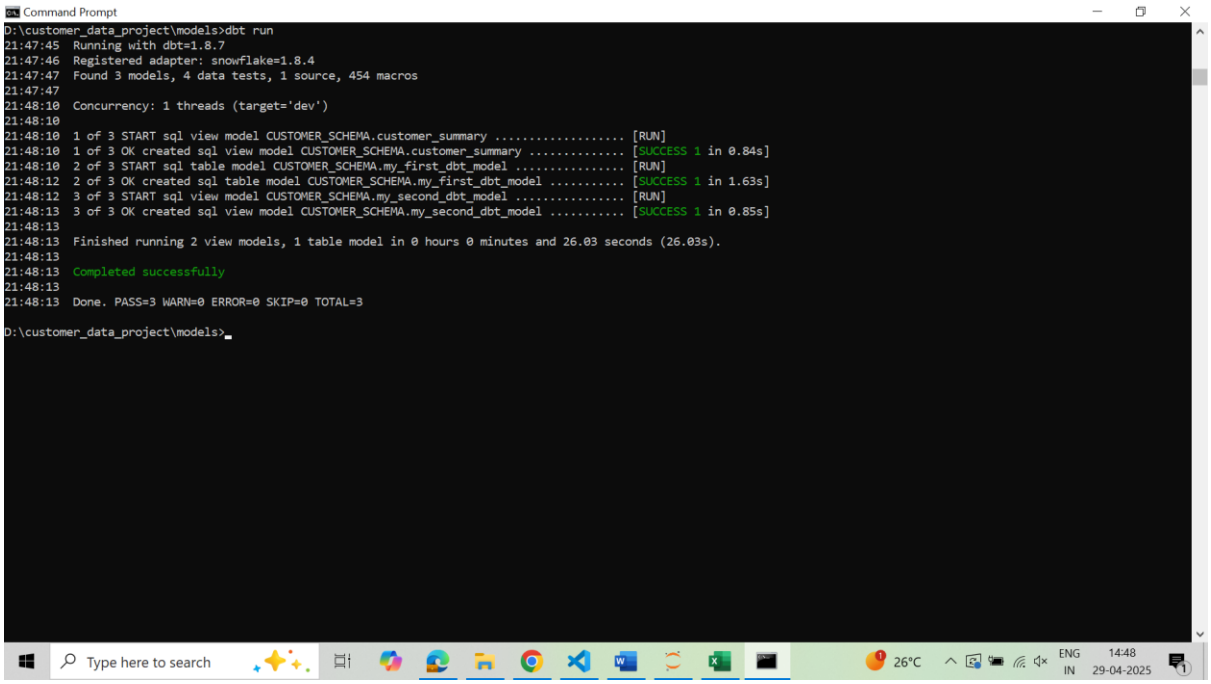




The below screenshot explains about the connection between DBT and Snowflake with the snowflake account credentials.

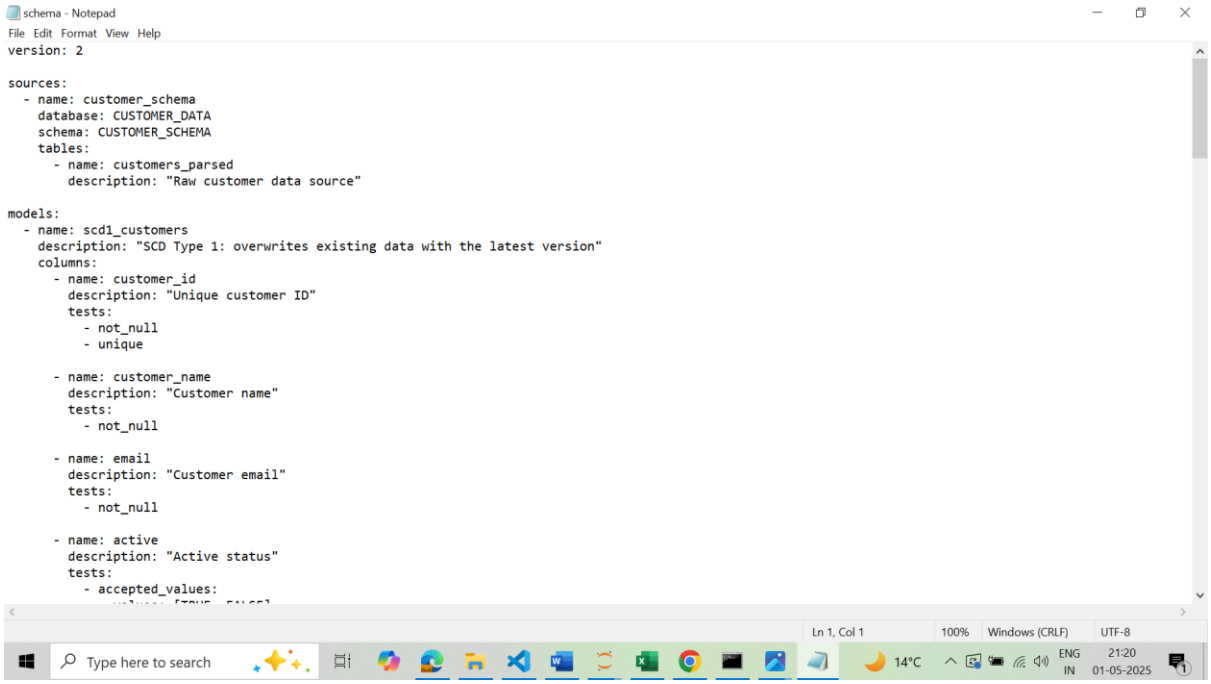



Command Prompt
21:15:13 Running with dbt=1.8.7
21:15:13 dbt version: 1.8.7
21:15:13 python version: 3.8.1
21:15:13 python path: c:\python\python38\python.exe
21:15:13 os info: Windows-10-10.0.19041-SP0
21:15:13 Using profiles dir at C:\Users\Srilakshmi N Murthy\.dbt
21:15:13 Using profiles.yml file at C:\Users\Srilakshmi N Murthy\.dbt\profiles.yml
21:15:13 Using dbt_project.yml file at D:\customer_data_project\dbt_project.yml
21:15:13 adapter type: snowflake
21:15:13 adapter version: 1.8.4
21:15:14 Configuration:
21:15:14 profiles.yml file [OK found and valid]
21:15:14 dbt_project.yml file [OK found and valid]
21:15:14 Required dependencies:
21:15:14 - git [OK found]
21:15:14 Connection:
21:15:14 account: ozb70682
21:15:14 user: SrilakshmiN
21:15:14 database: CUSTOMER_DATA
21:15:14 warehouse: COMPUTE_WH
21:15:14 role: ACCOUNTADMIN
21:15:14 schema: CUSTOMER_SCHEMA
21:15:14 authenticator: None
21:15:14 oauth_client_id: None
21:15:14 query_tag: None
21:15:14 client_session_keep_alive: False
21:15:14 host: None
21:15:14 port: None
21:15:14 proxy_host: None
21:15:14 proxy_port: None
21:15:14 protocol: None
21:15:14 connect_retries: 1
21:15:14 connect_timeout: None
21:15:14 retry_on_database_errors: False
21:15:14 retry_all: False
21:15:14 insecure_mode: False
21:15:14 reuse_connections: None
21:15:14 Registered adapter: snowflake=1.8.4
21:15:15 Connection test: [OK connection ok]
Type here to search
Chat...
ENG
IN
14:15
29-04-2025


Command Prompt
D:\customer_data_project\models>dbt run
21:47:45 Running with dbt=1.8.7
21:47:46 Registered adapter: snowflake=1.8.4
21:47:47 Found 3 models, 4 data tests, 1 source, 454 macros
21:47:47
21:48:10 Concurrency: 1 threads (target='dev')
21:48:10
21:48:10 1 of 3 START sql view model CUSTOMER_SCHEMA.customer_summary ..... [RUN]
21:48:10 1 of 3 OK created sql view model CUSTOMER_SCHEMA.customer_summary ..... [SUCCESS 1 in 0.84s]
21:48:10 2 of 3 START sql table model CUSTOMER_SCHEMA.my_first_dbt_model ..... [RUN]
21:48:12 2 of 3 OK created sql table model CUSTOMER_SCHEMA.my_first_dbt_model ..... [SUCCESS 1 in 1.63s]
21:48:12 3 of 3 START sql view model CUSTOMER_SCHEMA.my_second_dbt_model ..... [RUN]
21:48:13 3 of 3 OK created sql view model CUSTOMER_SCHEMA.my_second_dbt_model ..... [SUCCESS 1 in 0.85s]
21:48:13
21:48:13 Finished running 2 view models, 1 table model in 0 hours 0 minutes and 26.03 seconds (26.03s).
21:48:13
21:48:13 Completed successfully
21:48:13
21:48:13 Done. PASS=3 WARN=0 ERROR=0 SKIP=0 TOTAL=3
D:\customer_data_project\models>
Type here to search
26°C
ENG
IN
14:48
29-04-2025

Schema.yml file

The .yml (YAML) file in dbt is used to **document models, columns, and tests** in a structured format. It allows to **add descriptions for each model and column** and define **data quality tests** such as not_null, unique, and accepted_values. The schema.yml file is used to organize tests for all SCD models and provide clear metadata about each table and its columns.



MODEL CREATIONS

SCD1 – SCD1 is used to handle changes in dimensional data where the old data is overwritten by the new data. For example, customer email id is updated to 'new_email@example.com' for customer id '6' in CUSTOMERS_PARSED table. When we run the dbt for 'scd1_customers' details are overwritten without storing any history. **SCD1 (Slowly Changing Dimension Type 1)** overwrites existing dimension data with the latest updates, ensuring that only **the current state is stored.**

```
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET EMAIL = 'new_email@example.com'
WHERE CUSTOMER_ID = '6';

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD1_CUSTOMERS
```

scd1_customers - Notepad

File Edit Format View Help

```
{{ config(
    materialized='incremental',
    unique_key='customer_id'
) }}

{% if is_incremental() %}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
),
existing as (
    select * from {{ this }}
),
final as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active
    from source s
    left join existing e on s.customer_id = e.customer_id
    where e.customer_id is null
       or md5(s.customer_name || s.email || s.join_date || s.active)
            != md5(e.customer_name || e.email || e.join_date || e.active)
)

select * from final

{% else %}

select
    customer_id,
    customer_name,
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

Type here to search 14°C ENG IN 21:21 01-05-2025

Customers_parsed table is updated.

app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query

2025-04-27 4:13pm 2025-04-27 10:26pm

Databases Worksheets

Pinned (1)

MY_SECOND_DBT_MODEL

Search objects

CUSTOMER_SCHEMA

Tables

CUSTOMERS_PARSED

CUSTOMERS_RAW

SCD1_CUSTOMERS

CUSTOMERS_PARSED 101 Rows

#	CUSTOMER_ID	NUMBER(38,0)
A	CUSTOMER_NAME	VARCHAR(16777216)
A	EMAIL	VARCHAR(16777216)
🕒	JOIN_DATE	DATE
0|1	ACTIVE	BOOLEAN

ACCOUNTADMIN • COMPUTE_WH (X-Small) Share

CUSTOMER_DATA.CUSTOMER_SCHEMA Settings Code Versions

```
197 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_RAW
198
199
200 SELECT *
201 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD1_CUSTOMERS
202
203 SELECT *
204 FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
205
206 UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
207 SET EMAIL = 'new_email@example.com'
208 WHERE CUSTOMER_ID = '6';
209
210
211
```

Results Chart

	number of rows updated	number of multi-joined rows updated
1	1	0

Query Details

Query duration 827ms

Rows 1

Query ID 01bc0b3e-0105-1bad-0...

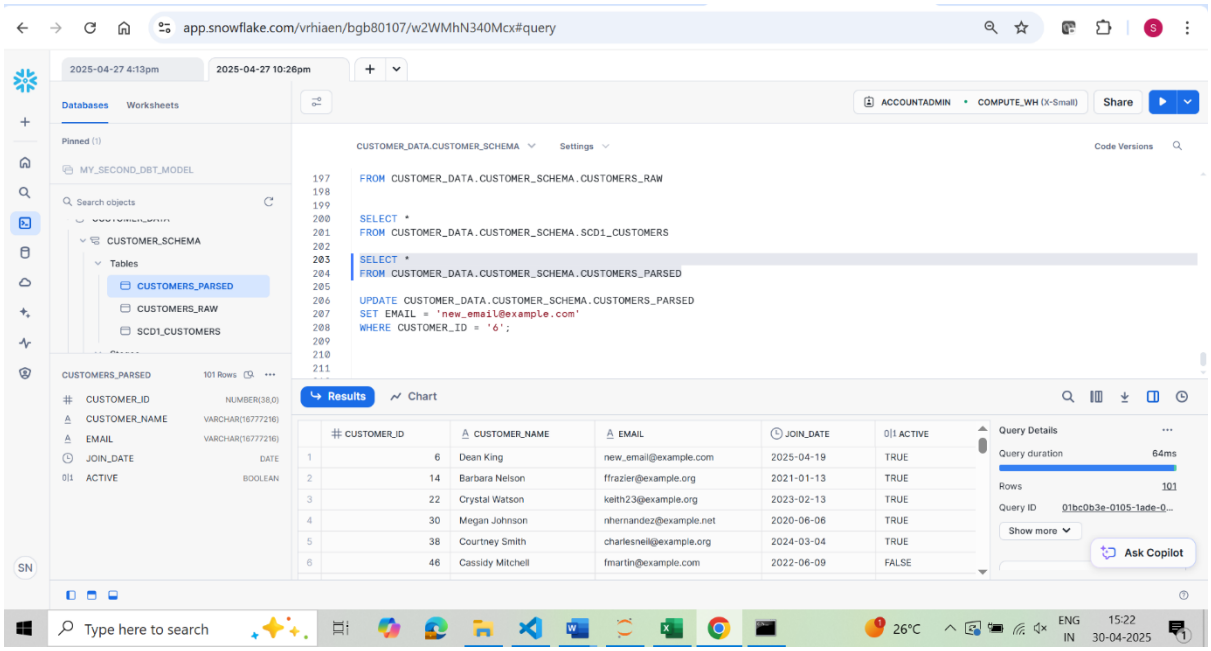
Show more

Ask Copilot

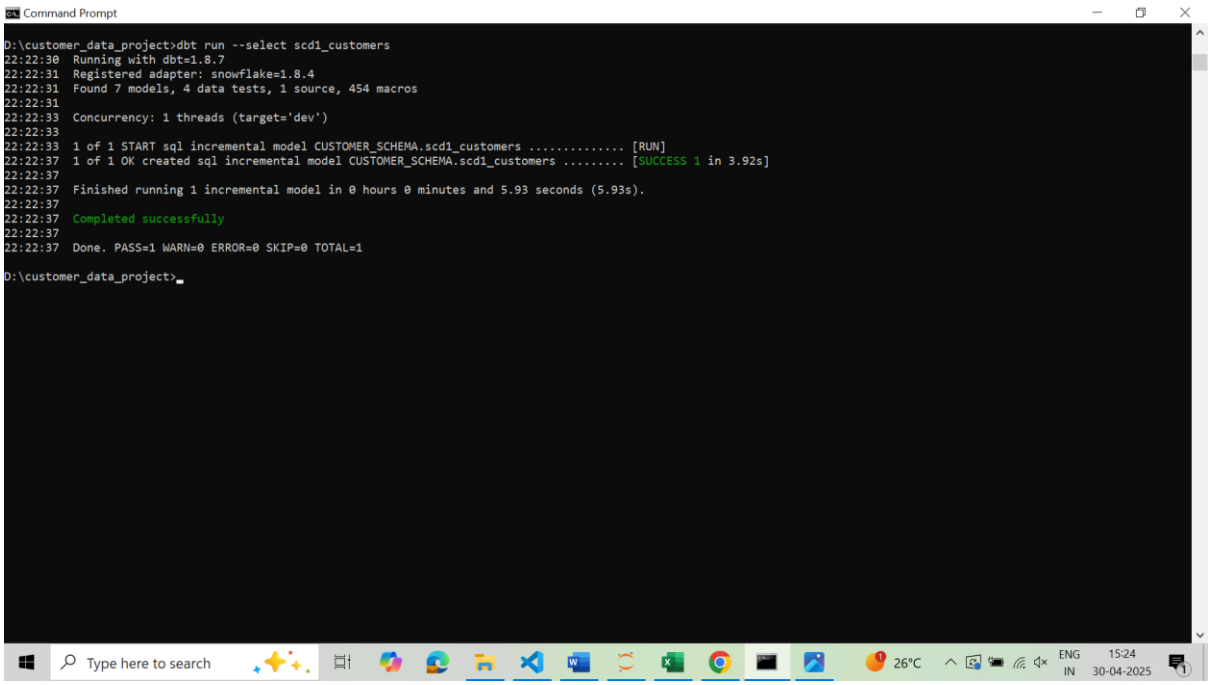
number of rows updated

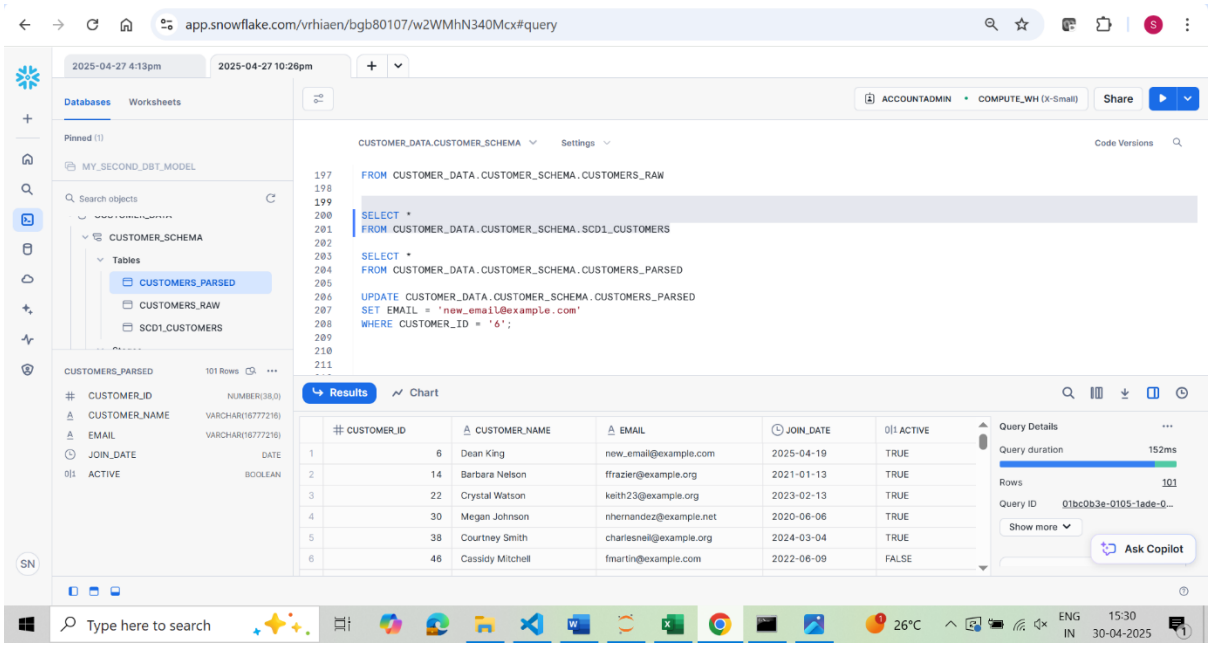
SN

Type here to search 26°C ENG IN 15:22 30-04-2025



Dbt is running for SCD1 Model



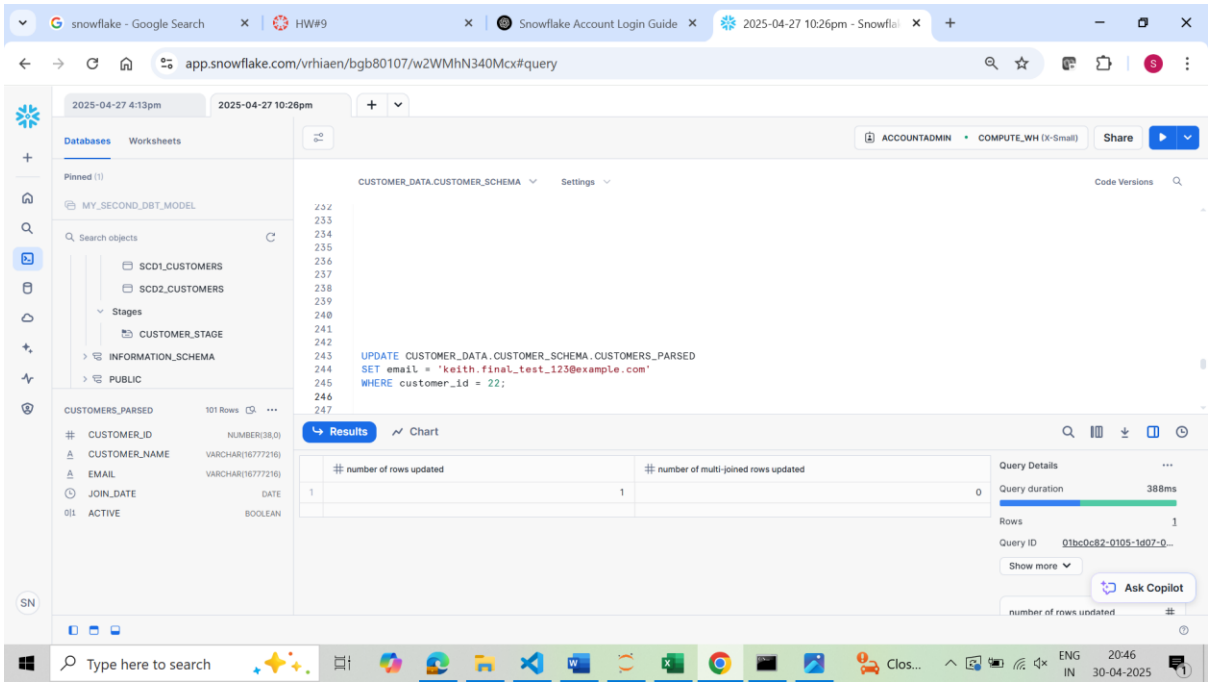


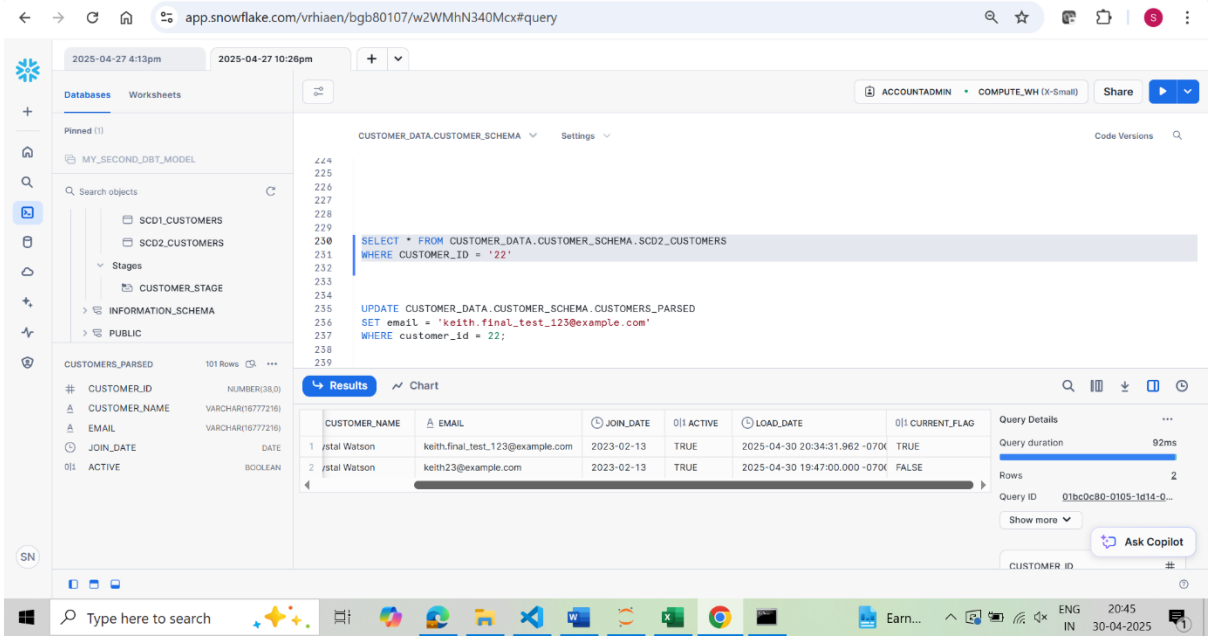
SCD2 - SCD2 (Slowly Changing Dimension Type 2) enables **full history tracking by inserting new rows** when changes occur, while keeping existing records unchanged. For example, customers email id is updated to **'keith.final_test_123@example.com' from 'keith23@example.com' for customer_id = '22'**. So SCD2 ensures to keep track of history by inserting new row by updating only that field. It keeps all others as it is. Hence it changes the **flag to 'FALSE' for the old record** and changes to **'TRUE' for new updated record**.

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'keith.final_test_123@example.com'
WHERE customer_id = 22;

SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD2_CUSTOMERS;

```
03:26:26    Database Error in model scd2_customers (models\scd2_customers.sql)
  002028 (42601): SQL compilation error:
  ambiguous column name 'LOAD_DATE'
  compiled code at target\run\customer_data_project\models\scd2_customers.sql
03:26:26
03:26:26  Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1

D:\customer_data_project>dbt run --select scd2_customers --full-refresh
03:29:12  Running with dbt=1.8.7
03:29:13  Registered adapter: snowflake=1.8.4
03:29:14  Found 8 models, 4 data tests, 1 source, 454 macros
03:29:14
03:29:15  Concurrency: 1 threads (target='dev')
03:29:15
03:29:15  1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd2_customers .............. [RUN]
03:29:17  1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd2_customers ......... [SUCCESS 1 in 1.91s]
03:29:17
03:29:17  Finished running 1 incremental model in 0 hours 0 minutes and 3.43 seconds (3.43s).
03:29:17
03:29:17  Completed successfully
03:29:17
```





```
  ambiguous column name 'LOAD_DATE'
  compiled code at target\run\customer_data_project\models\scd2_customers.sql
03:29:43
03:29:43  Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1

D:\customer_data_project>dbt run --select scd2_customers --full-refresh
03:34:28  Running with dbt=1.8.7
03:34:29  Registered adapter: snowflake=1.8.4
03:34:29  Found 8 models, 4 data tests, 1 source, 454 macros
03:34:29
03:34:31  Concurrency: 1 threads (target='dev')
03:34:31
03:34:31  1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd2_customers .............. [RUN]
03:34:33  1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd2_customers ......... [SUCCESS 1 in 1.96s]
03:34:33
03:34:33  Finished running 1 incremental model in 0 hours 0 minutes and 3.35 seconds (3.35s).
03:34:33
03:34:33  Completed successfully
03:34:33
03:34:33  Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
```

SCD3 - SCD3 is about tracking limited history by adding new columns rather than adding new rows. Instead of keeping **multiple rows per record** (like SCD2), SCD3 **stores the current value and one (or a few) previous values in additional columns.** For example, email is changed to **'scd3.newemail@example.com'** from **'nhernandez@example.com'.** It created a new column named 'prev_mail' and stored the history instead of creating new row. So **the row stays the same (no new row is added),** but the **columns are updated to reflect the change.**

UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'scd3.newemail@example.com'
WHERE customer_id = 30;

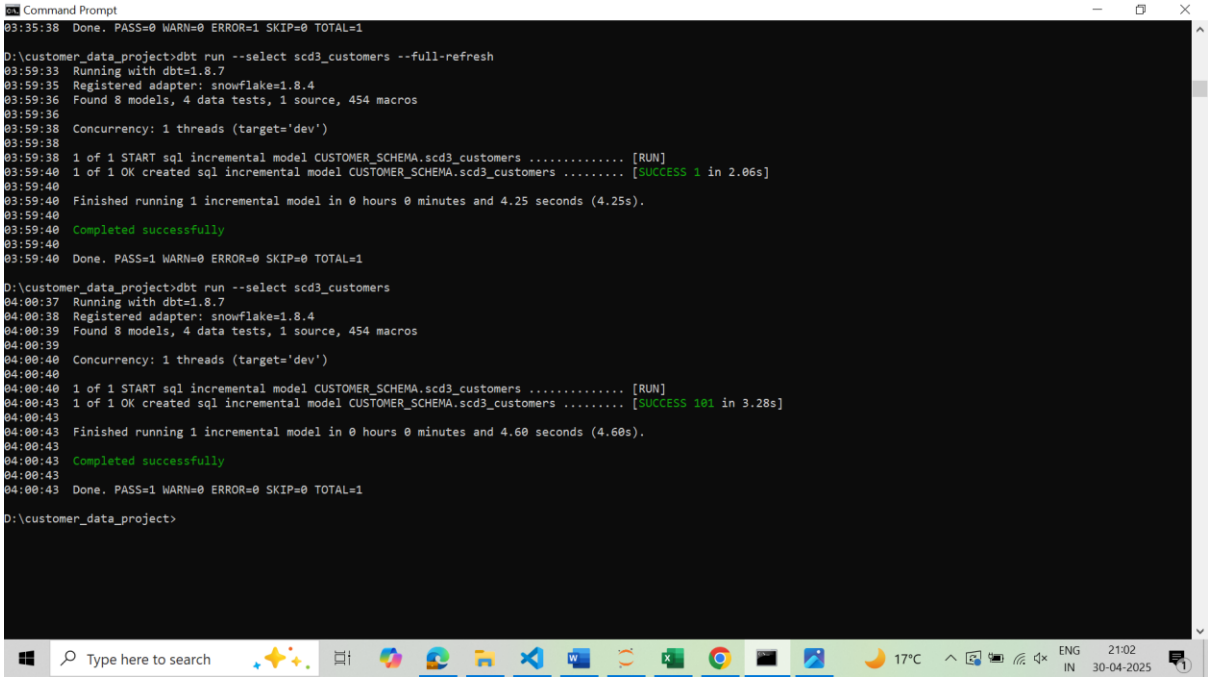
SELECT * FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD3_CUSTOMERS
WHERE customer_id = 30;

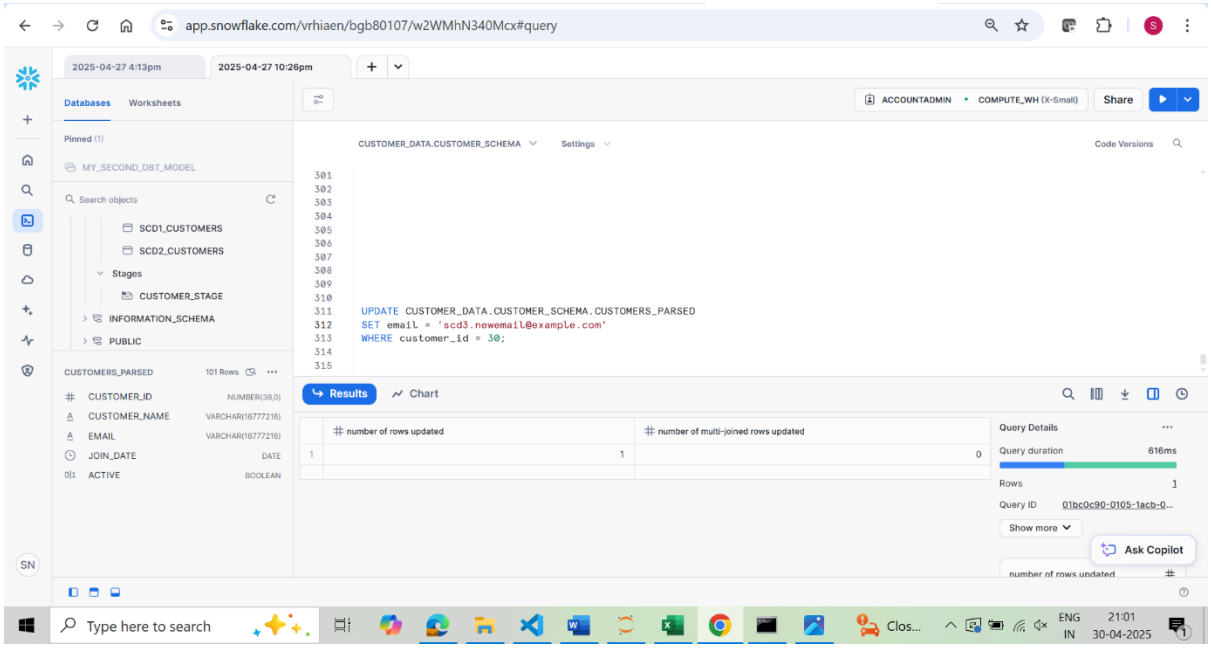
scd3_customers - Notepad

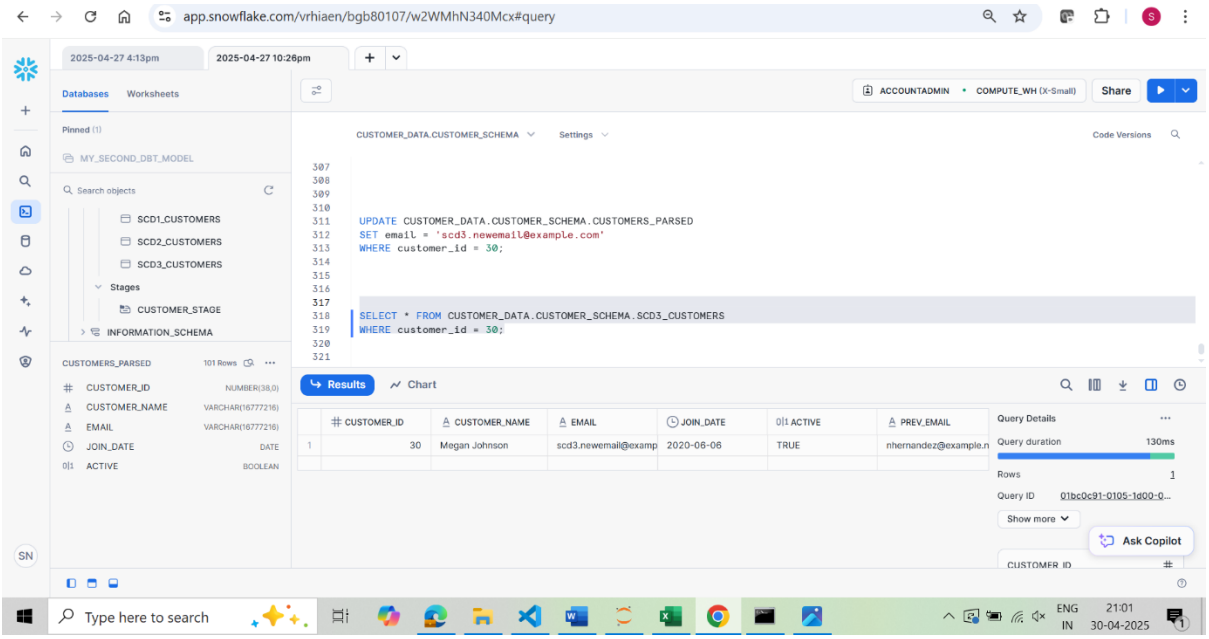
File Edit Format View Help

```
{{ config(
    materialized='incremental',
    unique_key='customer_id'
) }}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
)

{% if is_incremental() %}

, current_records as (
    select * from {{ this }}
)

, merged as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active,
        case
            when s.email != c.email then c.email
            else c.prev_email
        end as prev_email
    from source s
    left join current_records c
        on s.customer_id = c.customer_id
)

select * from merged

{% else %}
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8


Command Prompt
03:35:38 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd3_customers --full-refresh
03:59:33 Running with dbt=1.8.7
03:59:35 Registered adapter: snowflake=1.8.4
03:59:36 Found 8 models, 4 data tests, 1 source, 454 macros
03:59:38 Concurrency: 1 threads (target='dev')
03:59:38 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [RUN]
03:59:40 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [SUCCESS 1 in 2.06s]
03:59:40 Finished running 1 incremental model in 0 hours 0 minutes and 4.25 seconds (4.25s).
03:59:40 Completed successfully
03:59:40 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd3_customers
04:00:37 Running with dbt=1.8.7
04:00:38 Registered adapter: snowflake=1.8.4
04:00:39 Found 8 models, 4 data tests, 1 source, 454 macros
04:00:40 Concurrency: 1 threads (target='dev')
04:00:40 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [RUN]
04:00:43 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd3_customers ..... [SUCCESS 101 in 3.28s]
04:00:43 Finished running 1 incremental model in 0 hours 0 minutes and 4.60 seconds (4.60s).
04:00:43 Completed successfully
04:00:43 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>
Type here to search
17°C
ENG IN
21:02 30-04-2025

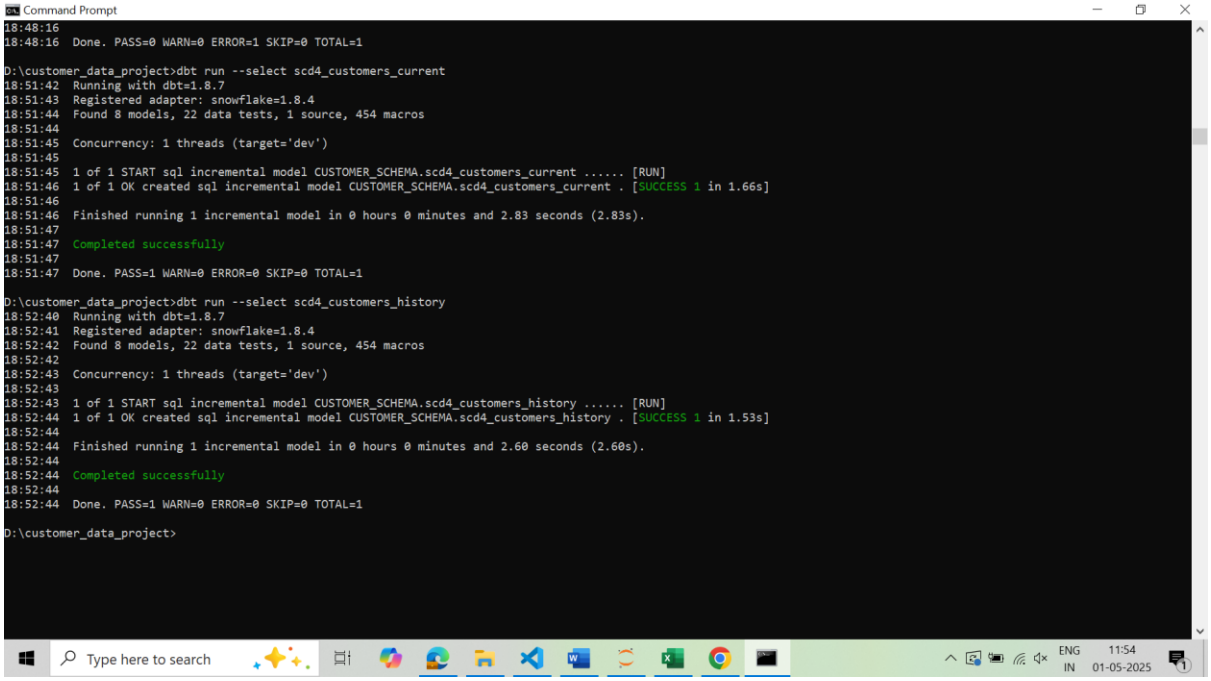

app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query
2025-04-27 4:13pm
2025-04-27 10:26pm
Databases
Worksheets
ACCOUNTADMIN • COMPUTE_WH (X-Small)
Share
CUSTOMER_DATA.CUSTOMER_SCHEMA
Settings
Code Versions
MY_SECOND_DBT_MODEL
SCD1_CUSTOMERS
SCD2_CUSTOMERS
Stages
CUSTOMER_STAGE
INFORMATION_SCHEMA
PUBLIC
CUSTOMERS_PARSED
101 Rows
CUSTOMER_ID NUMBER(38,0)
CUSTOMER_NAME VARCHAR(16777216)
EMAIL VARCHAR(16777216)
JOIN_DATE DATE
ACTIVE BOOLEAN
UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'scd3.newemail@example.com'
WHERE customer_id = 30;
Results
Chart
number of rows updated
number of multi-joined rows updated
Query Details
Query duration 616ms
Rows 1
Query ID 01bc0c90-0105-1acb-0...
Show more
Ask Copilot
Type here to search
ENG IN
21:01 30-04-2025

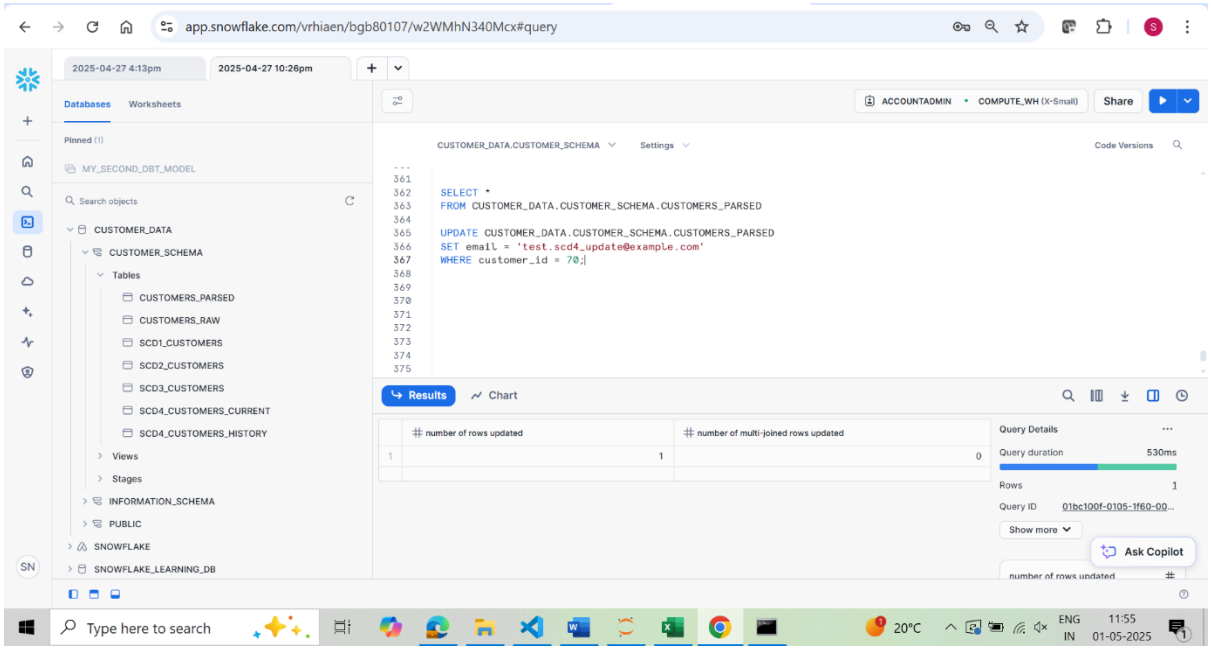


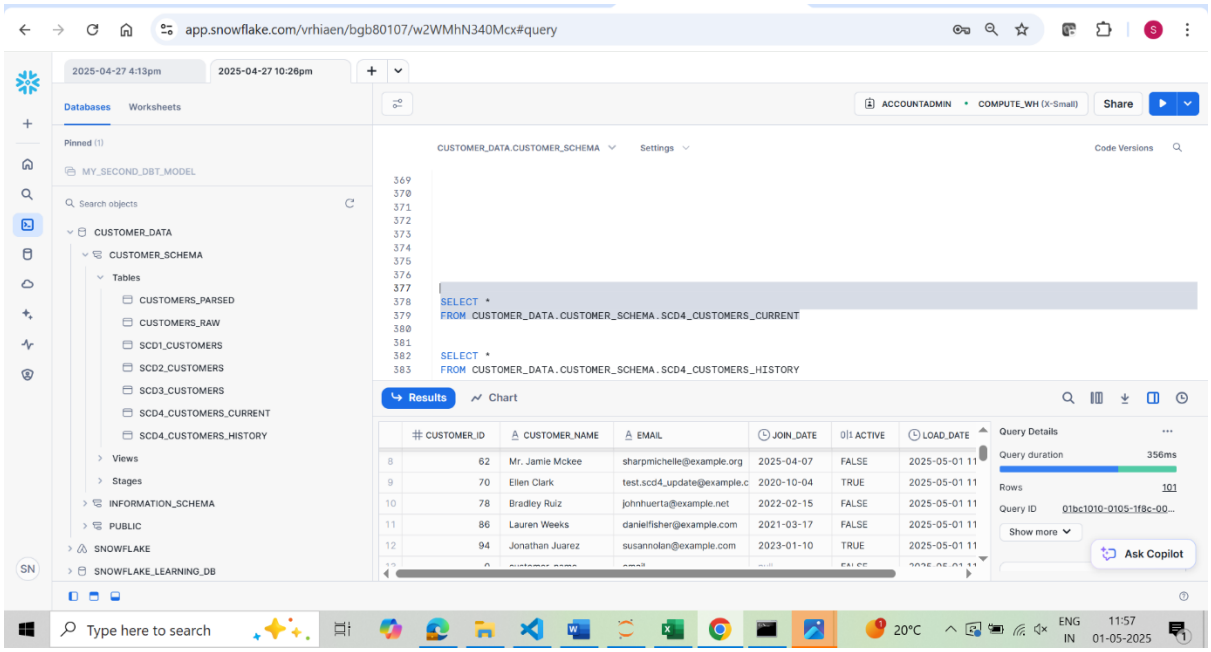
SCD4 - SCD4 is called a Hybrid approach because it **combines SCD1 (overwrite) with a separate history table.** Current data in one table is maintained(like SCD1 — overwrite the existing row). But changes are **archived into a separate history table** every time there's a change.

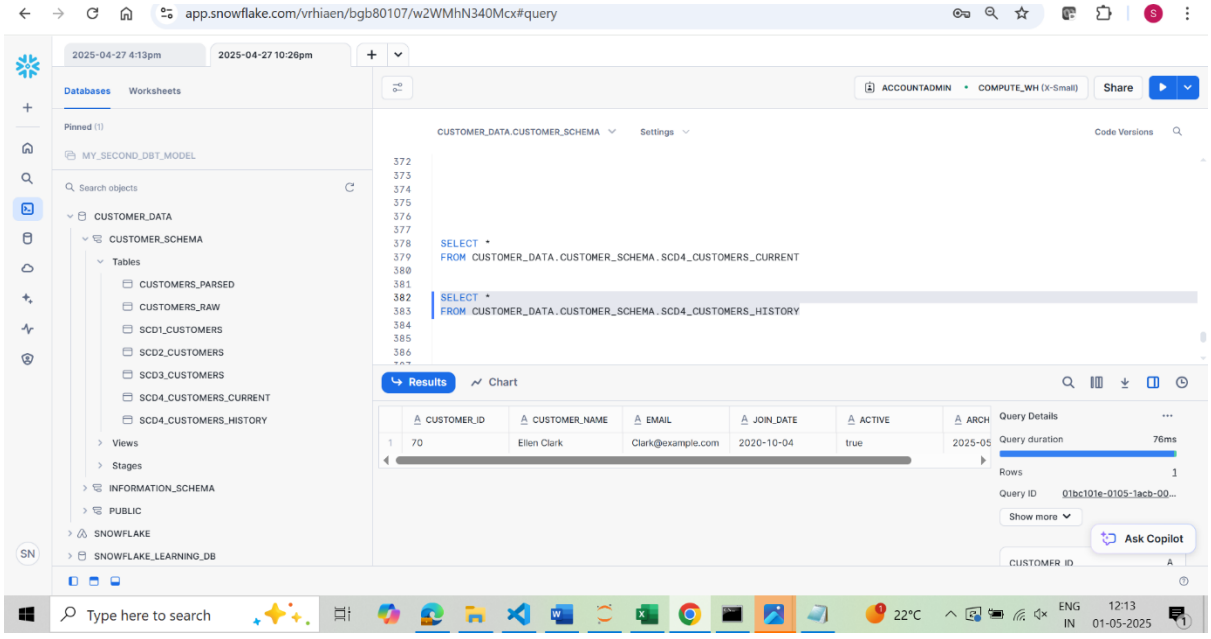
For example, customers email is updated to **'test.scd4_update@example.com'** in **CUSTOMERS_PARSED** from 'Clark@example.com'. It creates separate history table with the old data. While in current table, it retains the latest changes for the same record. The scd4_customers_current table stores only the latest version of each customer's record, similar to SCD1. The scd4_customers_history table captures every previous version of the customer's data before updates.

```
UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'test.scd4_update@example.com'
WHERE customer_id = 70;

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT

SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY
```


Command Prompt
18:48:16
18:48:16 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd4_customers_current
18:51:42 Running with dbt=1.8.7
18:51:43 Registered adapter: snowflake=1.8.4
18:51:44 Found 8 models, 22 data tests, 1 source, 454 macros
18:51:44
18:51:45 Concurrency: 1 threads (target='dev')
18:51:45
18:51:45 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd4_customers_current ..... [RUN]
18:51:46 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd4_customers_current . [SUCCESS 1 in 1.66s]
18:51:46
18:51:46 Finished running 1 incremental model in 0 hours 0 minutes and 2.83 seconds (2.83s).
18:51:47
18:51:47 Completed successfully
18:51:47
18:51:47 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>dbt run --select scd4_customers_history
18:52:40 Running with dbt=1.8.7
18:52:41 Registered adapter: snowflake=1.8.4
18:52:42 Found 8 models, 22 data tests, 1 source, 454 macros
18:52:42
18:52:43 Concurrency: 1 threads (target='dev')
18:52:43
18:52:43 1 of 1 START sql incremental model CUSTOMER_SCHEMA.scd4_customers_history ..... [RUN]
18:52:44 1 of 1 OK created sql incremental model CUSTOMER_SCHEMA.scd4_customers_history . [SUCCESS 1 in 1.53s]
18:52:44
18:52:44 Finished running 1 incremental model in 0 hours 0 minutes and 2.60 seconds (2.60s).
18:52:44
18:52:44 Completed successfully
18:52:44
18:52:44 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
D:\customer_data_project>
Type here to search
ENG IN
11:54
01-05-2025

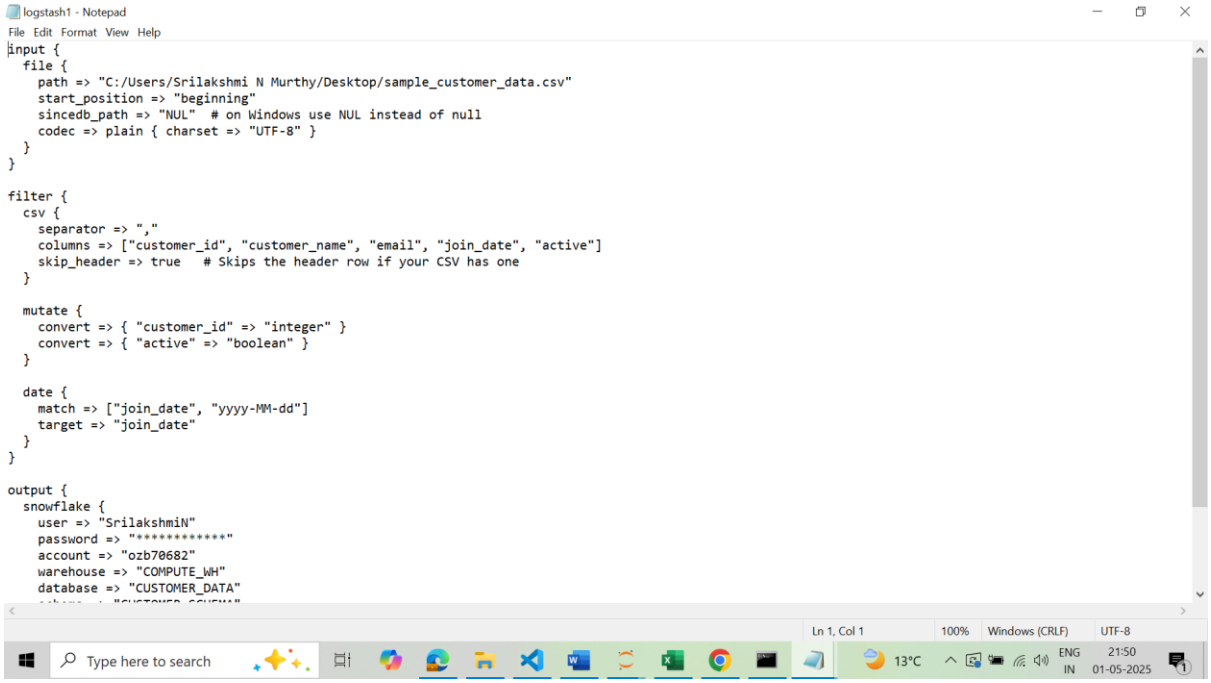

app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query
2025-04-27 4:13pm
2025-04-27 10:26pm
Databases
Worksheets
Pinned (1)
MY_SECOND_DBT_MODEL
Search objects
CUSTOMER_DATA
CUSTOMER_SCHEMA
Tables
CUSTOMERS_PARSED
CUSTOMERS_RAW
SCD1_CUSTOMERS
SCD2_CUSTOMERS
SCD3_CUSTOMERS
SCD4_CUSTOMERS_CURRENT
SCD4_CUSTOMERS_HISTORY
Views
Stages
INFORMATION_SCHEMA
PUBLIC
SNOWFLAKE
SNOWFLAKE_LEARNING_DB
ACCOUNTADMIN • COMPUTE_WH (X-Small)
Share
CUSTOMER_DATA.CUSTOMER_SCHEMA
Settings
Code Versions
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
UPDATE CUSTOMER_DATA.CUSTOMER_SCHEMA.CUSTOMERS_PARSED
SET email = 'test.scd4_update@example.com'
WHERE customer_id = 70;
Results
Chart
number of rows updated
number of multi-joined rows updated
1
0
Query Details
Query duration
530ms
Rows
1
Query ID
Show more
Ask Copilot
Type here to search
20°C
ENG IN
11:55
01-05-2025


app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query
2025-04-27 4:13pm
2025-04-27 10:26pm
Databases
Worksheets
ACCOUNTADMIN • COMPUTE_WH (X-Small)
Share
CUSTOMER_DATA.CUSTOMER_SCHEMA
Settings
Code Versions
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY
Results
Chart
CUSTOMER_ID
CUSTOMER_NAME
EMAIL
JOIN_DATE
ACTIVE
LOAD_DATE
62 Mr. Jamie Mckee sharpmichelle@example.org 2025-04-07 FALSE 2025-05-01 11
70 Ellen Clark test.scd4_update@example.c 2020-10-04 TRUE 2025-05-01 11
78 Bradley Ruiz johnhuerta@example.net 2022-02-15 FALSE 2025-05-01 11
86 Lauren Weeks danielfisher@example.com 2021-03-17 FALSE 2025-05-01 11
94 Jonathan Juarez susannolan@example.com 2023-01-10 TRUE 2025-05-01 11
Query Details
Query duration 356ms
Rows 101
Query ID 01bc1010-0105-1f8c-00...
Show more
Ask Copilot
Type here to search
20°C
ENG IN
11:57
01-05-2025


app.snowflake.com/vrhiaen/bgb80107/w2WMhN340Mcx#query
2025-04-27 4:13pm
2025-04-27 10:26pm
Databases
Worksheets
ACCOUNTADMIN • COMPUTE_WH (X-Small)
Share
CUSTOMER_DATA.CUSTOMER_SCHEMA
Settings
Code Versions
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_CURRENT
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.SCD4_CUSTOMERS_HISTORY
Results
Chart
CUSTOMER_ID
CUSTOMER_NAME
EMAIL
JOIN_DATE
ACTIVE
70 Ellen Clark Clark@example.com 2020-10-04 true
Query Details
Query duration 76ms
Rows 1
Query ID 01bc101e-0105-1acb-00...
Show more
Ask Copilot
Type here to search
22°C
ENG IN
12:13
01-05-2025

Error handling and Logging

1. In this project, proper error handling and logging are crucial for ensuring data quality and smooth troubleshooting. DBT automatically logs **detailed compilation, run, and test logs**. If any error occurs (e.g., SQL compilation errors, missing tables, type mismatches), DBT **captures and reports.**
2. Validation tests serves as automatic error detections. Any **unexpected nulls, duplicates, or invalid values** trigger clear test failures.
3. Incremental model safety has taken to avoid errors if any new changes are done in the records.
4. Error Prevention practices are used in code files, for example skip_header=>true to avoid misinterpreting CSV headers as data.
5. Logstash logging : Used sincedb_path => "NUL" and start_position => "beginning" to ensure no **duplicate processing errors** happen for repeated runs.



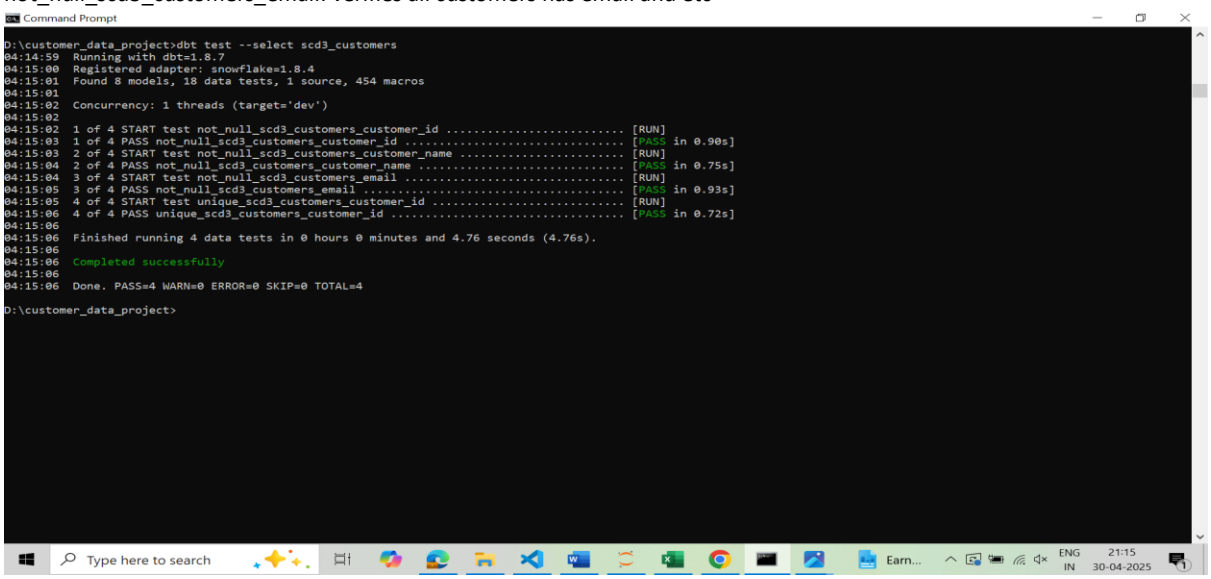
Below scripts shows all the test cases for scd1, scd2, scd3, scd4. It represents the data tests for particular fields in those particular SCDs.

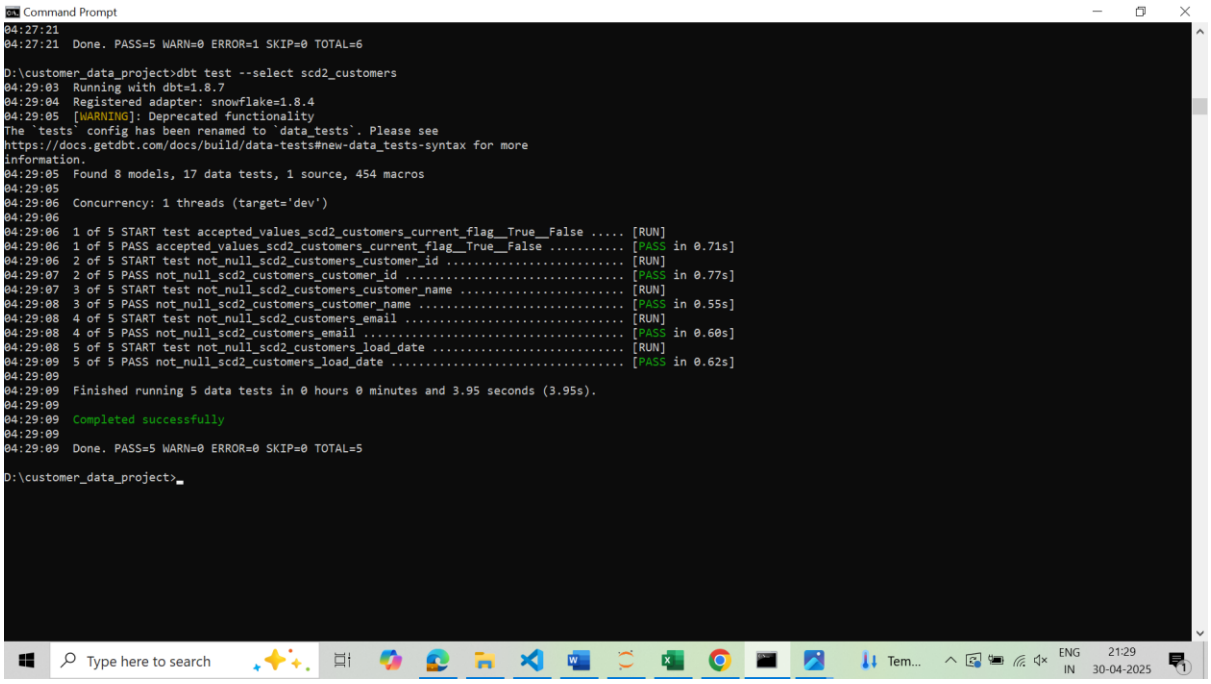
DBTs built in test cases

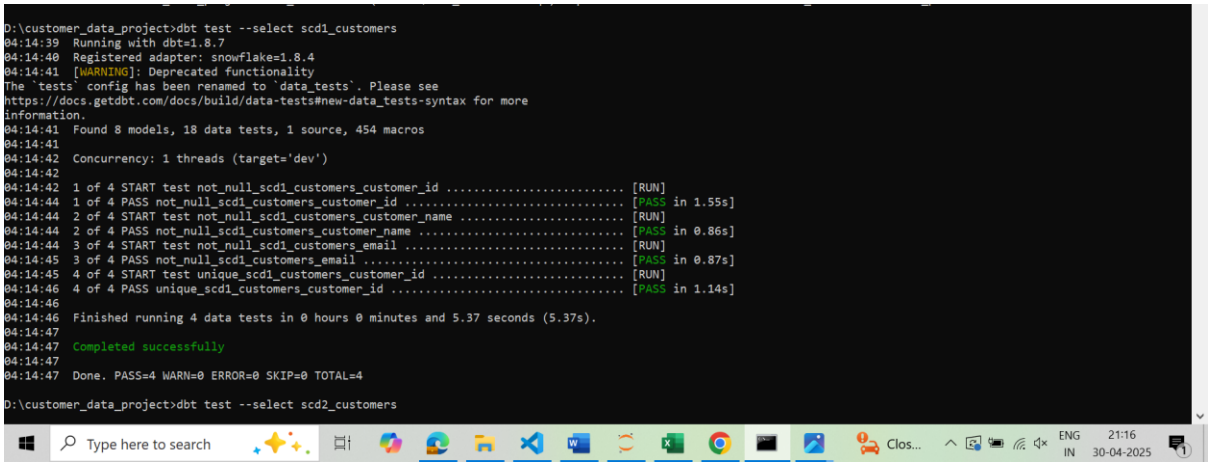
not_null_scd3_customers_customer_id : Ensures every customer has customer id

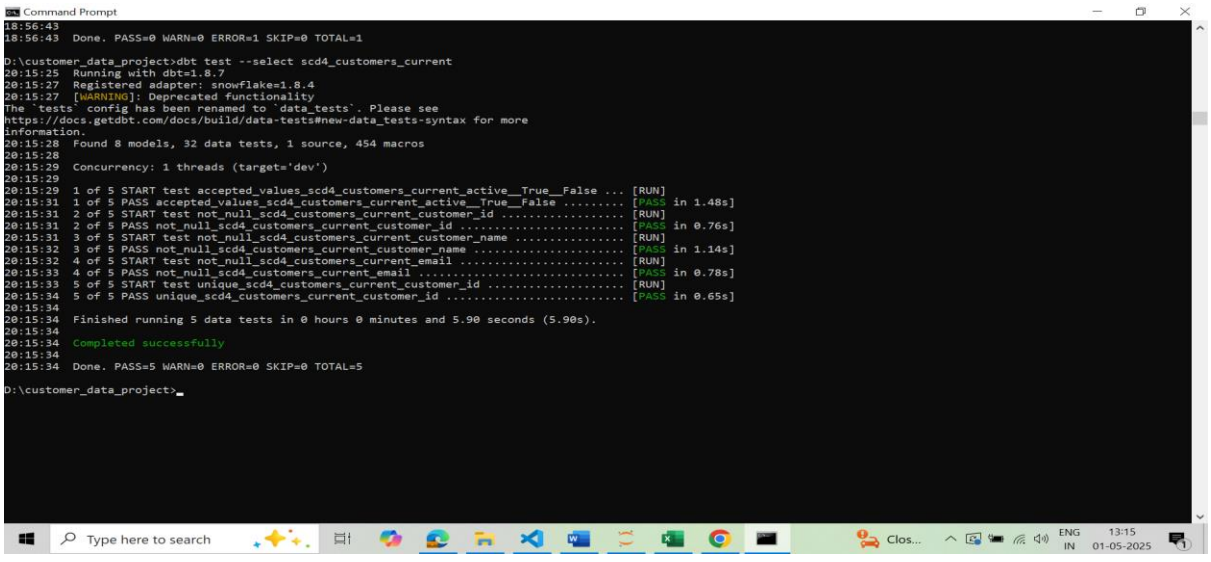
not_null_scd3_customers_customer_name: Confirms all customers has names

not_null_scd3_customers_email: verifies all customers has email and etc



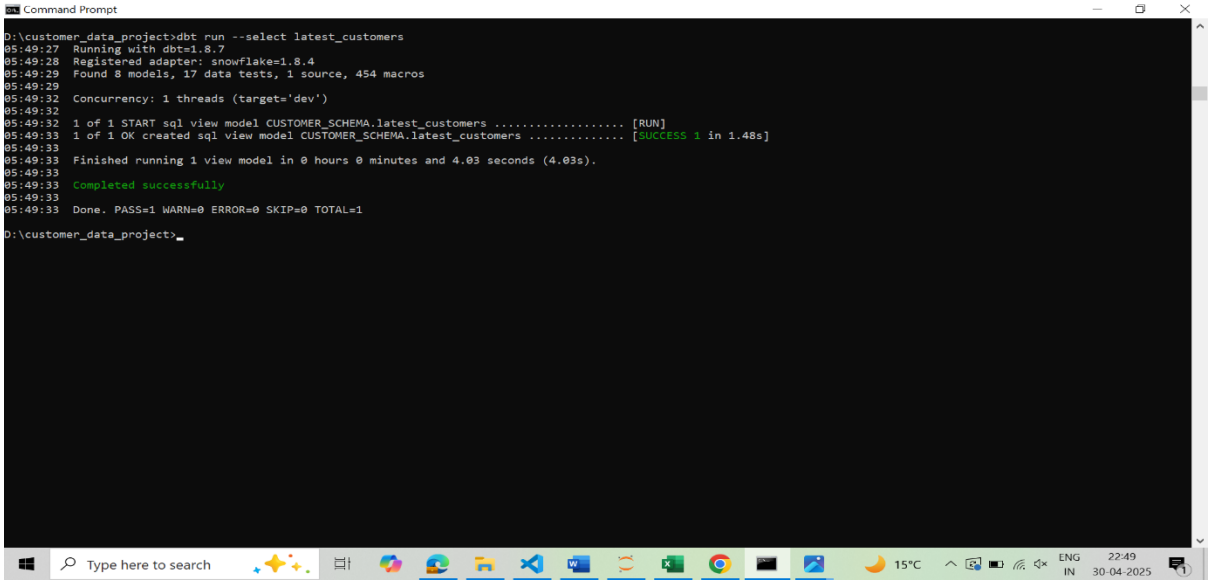

Command Prompt
04:27:21
04:27:21 Done. PASS=5 WARN=0 ERROR=1 SKIP=0 TOTAL=6
D:\customer_data_project>dbt test --select scd2_customers
04:29:03 Running with dbt=1.8.7
04:29:04 Registered adapter: snowflake=1.8.4
04:29:05 [WARNING]: Deprecated functionality
The `tests` config has been renamed to `data_tests`. Please see
https://docs.getdbt.com/docs/build/data-tests#new-data_tests-syntax for more
information.
04:29:05 Found 8 models, 17 data tests, 1 source, 454 macros
04:29:05
04:29:06 Concurrency: 1 threads (target='dev')
04:29:06
04:29:06 1 of 5 START test accepted_values_scd2_customers_current_flag__True__False ..... [RUN]
04:29:06 1 of 5 PASS accepted_values_scd2_customers_current_flag__True__False ..... [PASS in 0.71s]
04:29:06 2 of 5 START test not_null_scd2_customers_customer_id ..... [RUN]
04:29:07 2 of 5 PASS not_null_scd2_customers_customer_id ..... [PASS in 0.77s]
04:29:07 3 of 5 START test not_null_scd2_customers_customer_name ..... [RUN]
04:29:08 3 of 5 PASS not_null_scd2_customers_customer_name ..... [PASS in 0.55s]
04:29:08 4 of 5 START test not_null_scd2_customers_email ..... [RUN]
04:29:08 4 of 5 PASS not_null_scd2_customers_email ..... [PASS in 0.60s]
04:29:08 5 of 5 START test not_null_scd2_customers_load_date ..... [RUN]
04:29:09 5 of 5 PASS not_null_scd2_customers_load_date ..... [PASS in 0.62s]
04:29:09
04:29:09 Finished running 5 data tests in 0 hours 0 minutes and 3.95 seconds (3.95s).
04:29:09
04:29:09 Completed successfully
04:29:09
04:29:09 Done. PASS=5 WARN=0 ERROR=0 SKIP=0 TOTAL=5
D:\customer_data_project>
Type here to search
ENG
IN
21:29
30-04-2025

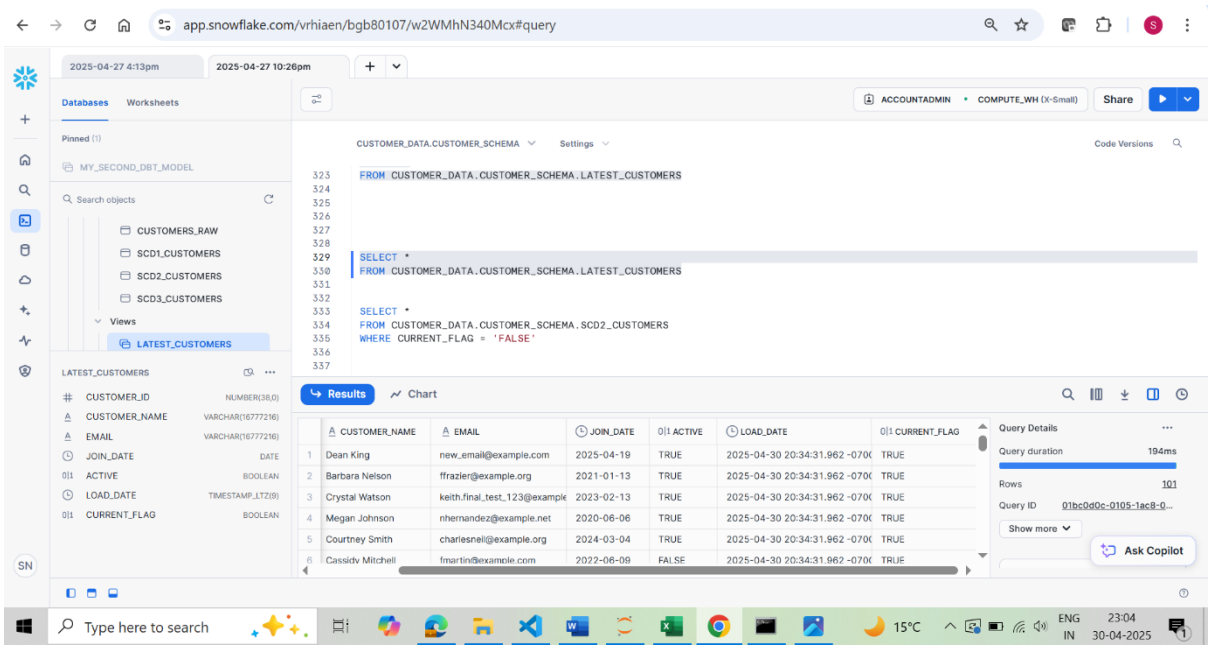

D:\customer_data_project>dbt test --select scd1_customers
04:14:39 Running with dbt=1.8.7
04:14:40 Registered adapter: snowflake=1.8.4
04:14:41 [WARNING]: Deprecated functionality
The `tests` config has been renamed to `data_tests`. Please see
https://docs.getdbt.com/docs/build/data-tests#new-data_tests-syntax for more
information.
04:14:41 Found 8 models, 18 data tests, 1 source, 454 macros
04:14:41
04:14:42 Concurrency: 1 threads (target='dev')
04:14:42
04:14:42 1 of 4 START test not_null_scd1_customers_customer_id ..... [RUN]
04:14:44 1 of 4 PASS not_null_scd1_customers_customer_id ..... [PASS in 1.55s]
04:14:44 2 of 4 START test not_null_scd1_customers_customer_name ..... [RUN]
04:14:44 2 of 4 PASS not_null_scd1_customers_customer_name ..... [PASS in 0.86s]
04:14:44 3 of 4 START test not_null_scd1_customers_email ..... [RUN]
04:14:45 3 of 4 PASS not_null_scd1_customers_email ..... [PASS in 0.87s]
04:14:45 4 of 4 START test unique_scd1_customers_customer_id ..... [RUN]
04:14:46 4 of 4 PASS unique_scd1_customers_customer_id ..... [PASS in 1.14s]
04:14:46
04:14:46 Finished running 4 data tests in 0 hours 0 minutes and 5.37 seconds (5.37s).
04:14:47
04:14:47 Completed successfully
04:14:47
04:14:47 Done. PASS=4 WARN=0 ERROR=0 SKIP=0 TOTAL=4
D:\customer_data_project>dbt test --select scd2_customers
Type here to search
ENG
IN
21:16
30-04-2025

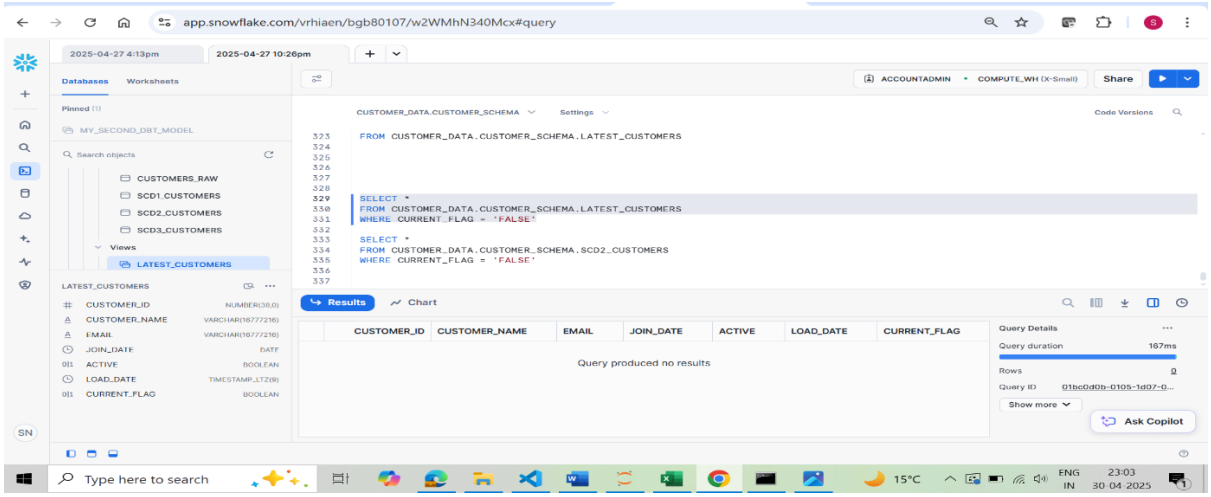

Command Prompt
18:56:43
18:56:43 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
D:\customer_data_project>dbt test --select scd4_customers_current
20:15:25 Running with dbt=1.8.7
20:15:27 Registered adapter: snowflake=1.8.4
20:15:27 [WARNING]: Deprecated functionality
The `tests` config has been renamed to `data_tests`. Please see
https://docs.getdbt.com/docs/build/data-tests#new-data_tests-syntax for more
information.
20:15:28 Found 8 models, 32 data tests, 1 source, 454 macros
20:15:28
20:15:29 Concurrency: 1 threads (target='dev')
20:15:29
20:15:29 1 of 5 START test accepted_values_scd4_customers_current_active__True__False ... [RUN]
20:15:31 1 of 5 PASS accepted_values_scd4_customers_current_active__True__False ..... [PASS in 1.48s]
20:15:31 2 of 5 START test not_null_scd4_customers_current_customer_id ..... [RUN]
20:15:31 2 of 5 PASS not_null_scd4_customers_current_customer_id ..... [PASS in 0.76s]
20:15:31 3 of 5 START test not_null_scd4_customers_current_customer_name ..... [RUN]
20:15:32 3 of 5 PASS not_null_scd4_customers_current_customer_name ..... [PASS in 1.14s]
20:15:32 4 of 5 START test not_null_scd4_customers_current_email ..... [RUN]
20:15:33 4 of 5 PASS not_null_scd4_customers_current_email ..... [PASS in 0.78s]
20:15:33 5 of 5 START test unique_scd4_customers_current_customer_id ..... [RUN]
20:15:34 5 of 5 PASS unique_scd4_customers_current_customer_id ..... [PASS in 0.65s]
20:15:34
20:15:34 Finished running 5 data tests in 0 hours 0 minutes and 5.90 seconds (5.90s).
20:15:34
20:15:34 Completed successfully
20:15:34
20:15:34 Done. PASS=5 WARN=0 ERROR=0 SKIP=0 TOTAL=5
D:\customer_data_project>
Type here to search
ENG
IN
13:15
01-05-2025

VIEW – A view named ‘latest_customers’ is created to keep track of latest records of only Active customers. When the below query is ran, we do not get any records as it keeps only ‘ACTIVE’ records in the view.

```
SELECT *
FROM CUSTOMER_DATA.CUSTOMER_SCHEMA.LATEST_CUSTOMERS
WHERE CURRENT_FLAG = 'FALSE'
```







Optimization Techniques Used

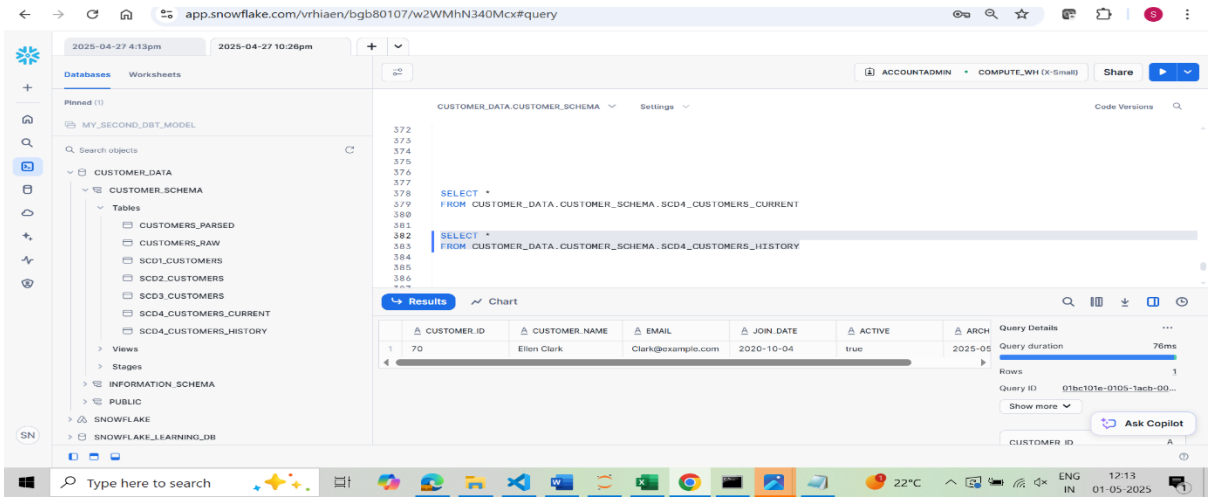
1.Incremental models: Incremental models are developed for the SCDs (is_incremental() blocks). This means **only new or changed records are processed** during subsequent runs**. This reduces execution time and warehouse cost significantly when compared to full table rebuilds.**

scd2_customers - Notepad

File Edit Format View Help

```
{{ config(
    materialized='incremental',
    unique_key='customer_id, load_date'
) }}

with source as (
    select * from {{ source('customer_schema', 'customers_parsed') }}
)

{% if is_incremental() %}

, current_customers as (
    select
        c.customer_id,
        c.customer_name,
        c.email,
        c.join_date,
        c.active,
        c.load_date,
        c.current_flag
    from {{ this }} as c
    where c.current_flag = true
```

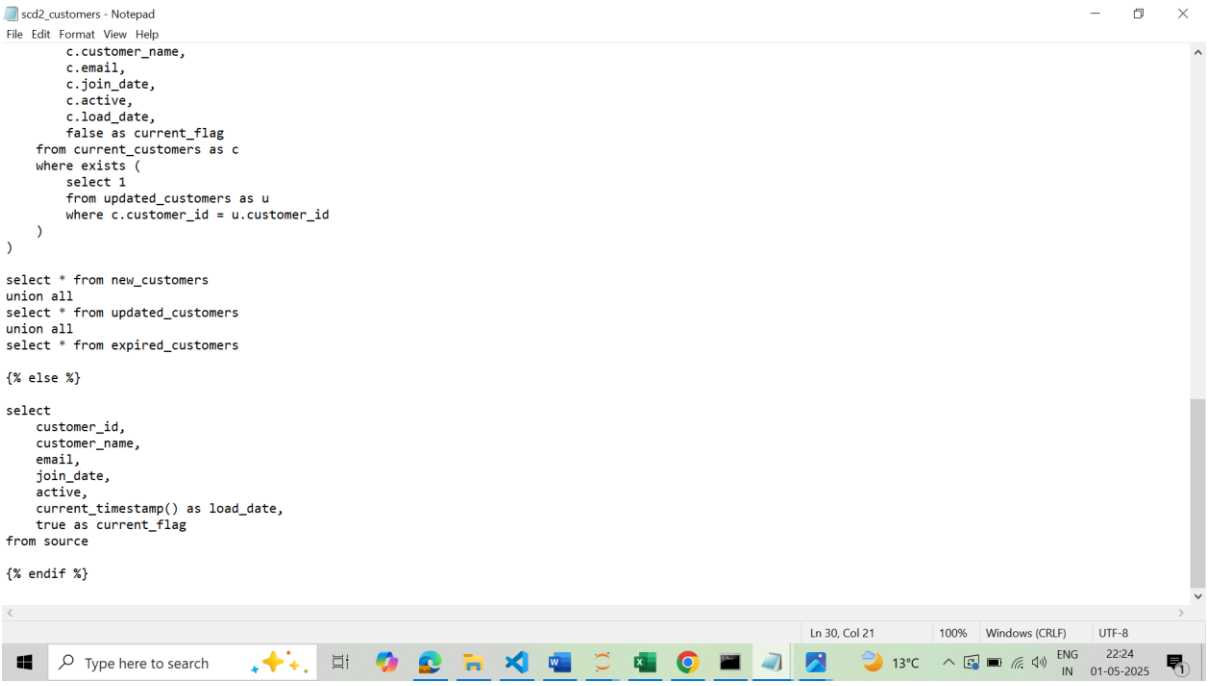
2.Source filtering – In SCD SQL logic, source data is filtered to include only new records or history records based on SCD logic. For example, SCD4_history stores history of only changed rows rather than storing whole table



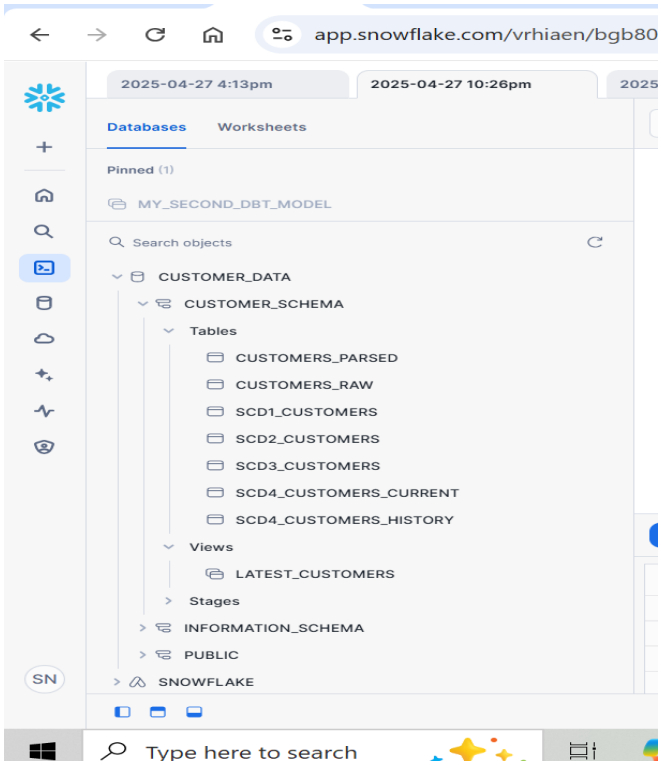
3.**Efficient joins and Indexing –** To keep the queries simple, only simple and fewer joins are used in SCDs to improve performance. There are no extra cross joins. They are very limited to reduce costs on joins.

```
        c.join_date,
        c.active,
        c.load_date,
        c.current_flag
    from {{ this }} as c
    where c.current_flag = true
)

, new_customers as (
    select
        s.customer_id,
        s.customer_name,
        s.email,
        s.join_date,
        s.active,
        current_timestamp() as load_date,
        true as current_flag
    from source as s
    left join current_customers as c
        on s.customer_id = c.customer_id
    where c.customer_id is null
)
```

4. **Modular Query Structure –** SQL scripts are designed such a way as updated_customers, expired_customers, new_customers to **reuse logic efficiently** without redundancy. This improves readability and allows Snowflake's optimizer to execute CTEs more intelligently.



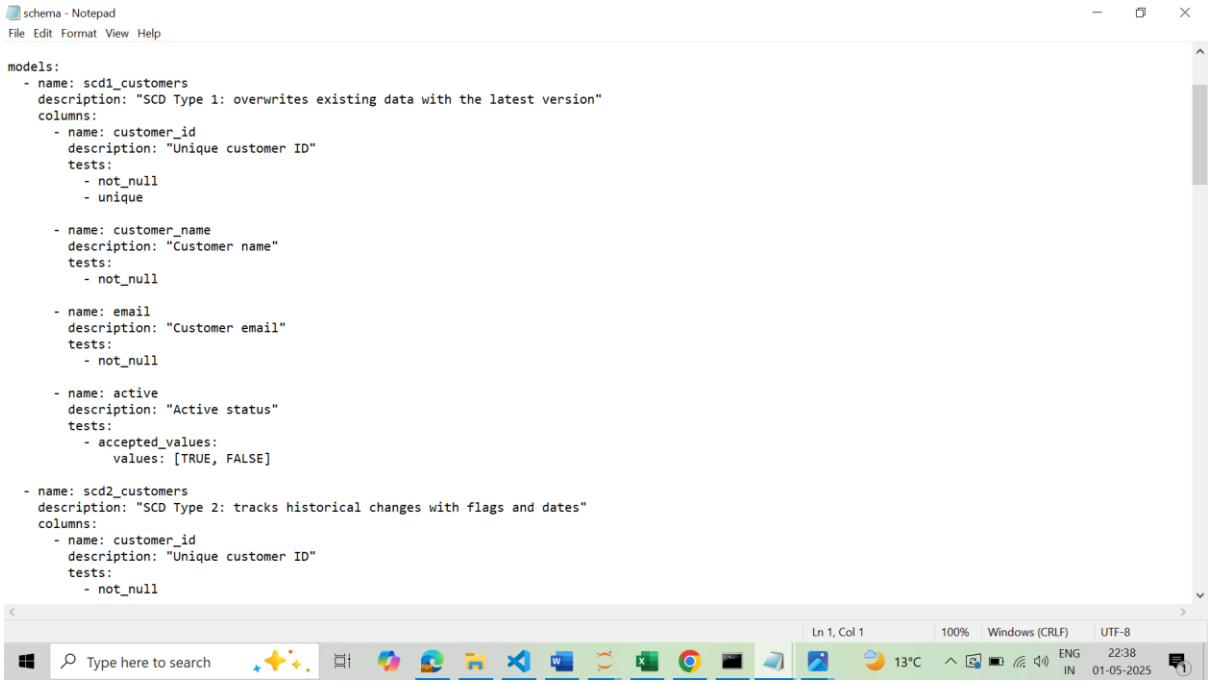
5.**DBT Materializations-** Separate models are created such as tables, views, incremental models based on use cases and faster access. For example, if they need only latest records, they can access latest_customers view. If they need only history, they can access scd4_history table.



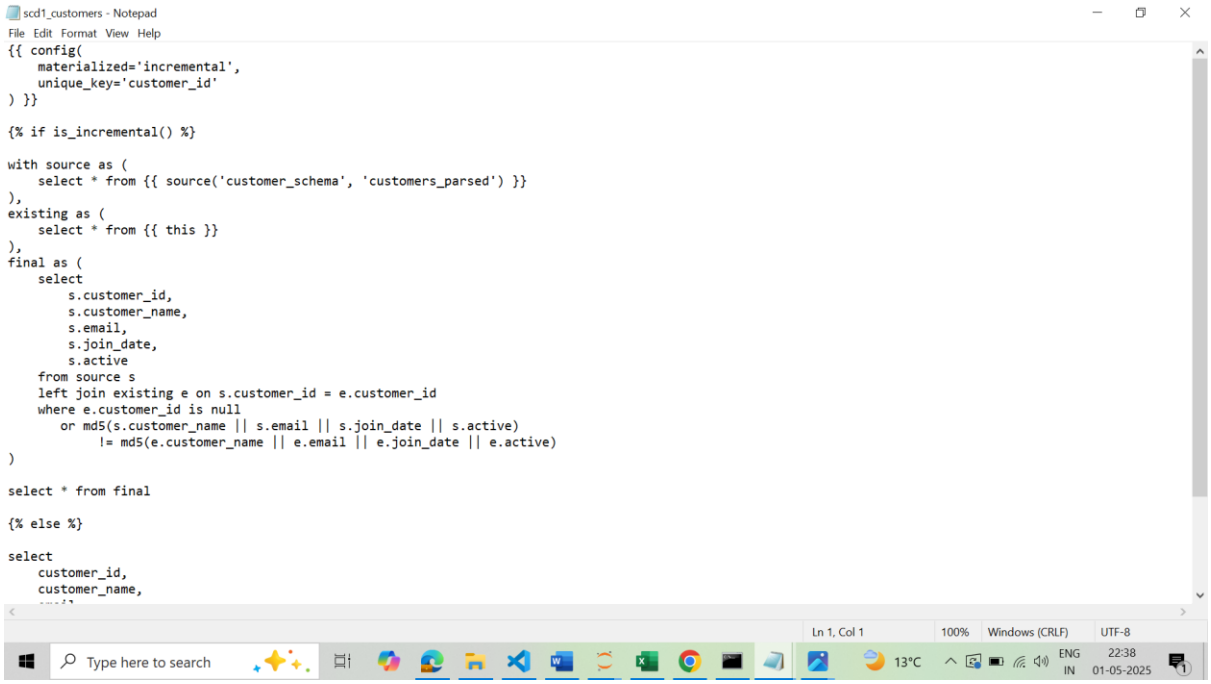
Code Reusability and Modularization

1.Reusable Models - Created **separate DBT models** for each SCD type (scd1_customers, scd2_customers, scd3_customers, scd4_customers_current, scd4_customers_history). Each model uses {{ ref() }} to **reference upstream models** instead of hardcoding table names, enabling easy reuse and dependency management. (Refer SCD scripts)

2.**YAML-Driven Documentation and Tests –** Created **single .yml file to organize column level** descriptions and reusable tests. Tests and docs are linked **to each model dynamically**, keeping code DRY (Don't Repeat Yourself).



3. **Configurable Materialization-** Leveraged DBT's {{ config() }} block to **parameterize materialization** (e.g., materialized='incremental'), allowing **reuse of the same pattern across models** without duplicating logic. This modular structure makes the pipeline **scalable and maintainable**—new dimensions or changes can be implemented by adding or updating **small, isolated modules** instead of touching everything.



4. Centralized Source Definition – Customers_parsed table is defined such that only once in .yml file. All other models pulls data from the same source file every time ensuring **no duplication and consistency** across all files.

5. Parameter-Driven Design- Introduced parameters such that future attributes changes can be added within same column with out duplicating another column. For example, if persons email id get changed twice, its get updated in that list rather than creating separate list.

Rubric Criteria	**STATUS**
Environment setup and repository configuration	DONE
Correct implementation of incremental loading -Create all models appropriately - Create mappings appropriately	DONE
Proper Error handling and Logging	DONE
Optimization techniques used	DONE
Code reusability and modularization	DONE
Documentation Quality	DONE